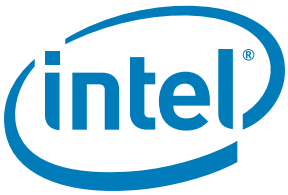


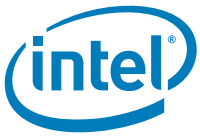
# Intel® Itanium® 2 Processor

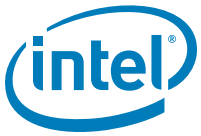
## Specification Update

***January 2007***

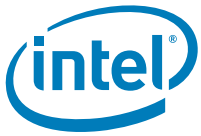
**Notice:** The Intel® Itanium® 2 processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Document Number: 251141-048





# Contents



# 1 Revision History

| Version | Description | Date |
|---|---|---|
| -048 | Added erratum E184, SDM Volume 2.2 documentation changes and a specification clarification | January 2007 |
| -047 | Added errata 177 through 183. Added IA-32 Execution Layer Errata 61 through 85. Added Itanium® 2 documentation change and specification clarification. Updated Fixed status for E124 - E165 in "Summary Table of Changes" for PAL 8.30 release for the Intel® Itanium® 2 9000 Series processor | December 2006 |
| -046 | Added errata 167 - 176 and updates to the Summary Table of Changes for errata fixed in the PAL 8.30 release for the Intel® Itanium® 2 9000 Series processor | November 2006 |
| -045 | Added errata 160 – 166 and added specification changes 3 – 5 for the Dual-Core Intel® Itanium® 2 Processor 9000 Series processor | October 2006 |
| -044 | Added errata 159 | September 2006 |
| -043 | Added errata 155-158, updated errata 110, 153 and 154, updated the Dual-Core Intel® Itanium® 2 processor 9000 series sku table, added PAL revision table 2.20 for Intel® Itanium® 2 Processor with up to 9 MB L3 Cache | August 2006 |
| -042 | Added Dual-Core Intel® Itanium® 2 processor 9000 series errata table and errata write ups, and identification information | July 2006 |
| -041 | Updated erratum 100. Added errata 110-111. Added IA-32 execution layer errata 45-60 | June 2006 |
| -040 | Added IA-32 execution layer version 6.5; added IA-32 execution layer errata 41-44; updated IA-32 execution layer summary table | January 2006 |
| -039 | Updated status for IA-32 execution layer errata 24-36 | December 2005 |
| -038 | Added IA-32 execution layer errata 38-40 | November 2005 |
| -037 | Added IA-32 execution layer errata 24-37 | October 2005 |
| -036 | Add PAL version 2.15, 5.73 and 7.79 | August 2005 |
| -035 | Added errata 108-109; added Intel® Itanium® 2 Processor with 1.66 GHz with up to 9 MB L3 Cache; added PAL version 2.14; added S-Spec numbers SL8JK and SL8JJ | July 2005 |
| -034 | Updated IA-32 execution layer erratum 1; added IA-32 execution layer errata 20-23 | June 2005 |
| -033 | Added erratum 107; added Intel® Itanium® 2 Processor (up to 9 MB L3 cache) A2 stepping and mixed stepping statement; added PAL versions 5.72 and 7.78; added IA-32 Execution Layer Specification Clarification 14; updated IA-32 Execution Layer Specification Clarification 2 | May 2005 |
| -032 | Added erratum #106; updated erratum 103; added PAL version 2.10 | April 2005 |
| -031 | Added errata 104-105; added Specification Clarification 7 | March 2005 |
| -030 | Updated IA-32 Execution Layer erratum 1; added IA-32 Execution Layer errata 18-19; added IA-32 Execution Layer Specification Clarifications 12-13. | February 2005 |
| -029 | Added errata 102-103; added IA-32 Execution Layer version 5.3; added Specification Changes 1-5; added Specification Clarifications 3-6 | January 2005 |
| -028 | Added Intel® Itanium® 2 Processor with 1.60 GHz with up to 9 MB L3 Cache, Low Voltage Intel® Itanium® 2 Processor with 1.30 GHz with 3 MB L3 Cache and Intel® Itanium® 2 Processor with 1.50 GHz with 4 MB L3 Cache to Table 3-1; added S-Spec numbers: SL87H, SL7EB, SL7EC, SL7ED, SL7EF and SL7SD; added PAL version 1.27; added errata 98-101; added Intel® Itanium® 2 Processor (up to 3 MB/6 MB L3 cache) Specification Clarification 2 | November 2004 |
| -027 | Added IA-32 Execution Layer version 4.4 | September 2004 |

| Version | Description | Date |
|---|---|---|
| -026 | Added Intel® Itanium® 2 Processor (up to 3 MB L3 cache) Specification Clarification 2 and Document Change 1; added Intel® Itanium® 2 Processor (up to 6 MB L3 cache) Specification Clarification 2 and Document Change 1 | August 2004 |
| -025 | Added PAL versions 7.77 and 5.69; updated workaround for erratum 61 | July 2004 |
| -024 | Updated workaround for erratum 61. | June 2004 |
| -023 | Added errata 94-97; added Intel® Itanium® 2 Processor with 1.60 GHz with 3 MB L3 Cache to Table 1-1; added S-spec number SL7FQ. | May 2004 |
| -022 | Added errata 92-93; added IA-32 execution layer erratum 17: Added Intel® Itanium® 2 Processor with 1.40 GHz with 3 MB L3 Cache to Table 1-1; added S-spec number SL7FP. | April 2004 |
| -021 | Added errata 88-91; added Intel® Itanium® 2 Processor (up to 3 MB L3 cache) PAL version 7.73 and Intel® Itanium® 2 Processor (up to 6 MB L3 cache) PAL version 5.65; added Intel® Itanium® 2 Processor (up to 3 MB L3 cache) Specification Clarification 1 and Intel® Itanium® 2 Processor (up to 6 MB L3 cache) Specification Clarification 1. | March 2004 |
| -020 | Added errata 83-87; added Intel® Itanium® 2 Processor (up to 3 MB L3 cache) PAL version 7.71 and Intel® Itanium® 2 Processor (up to 6 MB L3 cache) PAL version 5.61; updated workaround for erratum 61.<br>Updated problem and implication for IA-32 Execution Layer erratum 1; added IA-32 Execution Layer errata 2-16; added IA-32 Execution Layer Specification Clarifications 1-11. | January 2004 |
| -019 | Added errata 80-82. | December 2003 |
| -018 | Added errata 75-79. | November 2003 |
| -017 | Added errata 71-74. | October 2003 |
| -016 | Added errata 68-70; added Low Voltage Intel® Itanium® 2 Processor with 1.0 GHz with 1.5 MB L3 Cache and Intel® Itanium® 2 Processor with 1.40 GHz with 1.5 MB L3 Cache to Table 3-1; added S-Spec numbers SL76K and SL754; added *DP Optimized Intel® Itanium® 2 Processor Datasheet* to the list of Affected/Related Documents. | September 2003 |
| -015 | Added errata 65-67; updated the *Intel® Itanium® Architecture Software Developer's Manual Specification Update* document number in the list of Affected/Related Documents. | August 2003 |
| -014 | Added errata 61-62. | July 2003 |
| -013 | Added Intel® Itanium® 2 processor with 6 MB L3 cache information; added new errata summary tables and Table 3-1; removed Specification Clarification 1; removed Documentation Changes 1-2; added errata 59, 63-64. | June 2003 |
| -012 | Updated Implication for erratum 60. | June 2003 |
| -011 | Added erratum 60; removed erratum 59. | June 2003 |
| -010 | Added errata 55-59. | May 2003 |
| -009 | Added errata 53-54; added PAL version 7.40. | March 2003 |
| -008 | Updated workaround for erratum 48; added erratum 52; added PAL version 7.37. | February 2003 |
| -007 | Added errata 49-51; added Documentation Change 2. | January 2003 |
| -006 | Added errata 47-48. | December 2002 |
| -005 | Added errata 43-46; added PAL version 7.36. | November 2002 |
| -004 | Added errata 38-42. | October 2002 |
| -003 | Added errata 30-37; added PAL version 7.31; added Documentation Change 1; added Specification Clarification 1. | September 2002 |
| -002 | Added errata 20-29. | August 2002 |
| -001 | Initial release of this document. | July 2002 |

# 2 Preface

This document is an update to the specifications contained in the Affected/Related Documents table below. This document is a compilation of device and documentation errata, specification clarifications, and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

This document may also contain information that was not previously published.

## 2.1 Affected/Related Documents

| Title | Document # |
|---|---|
| *Intel® Itanium® 2 Processor Datasheet* | 250945 |
| *DP Optimized Intel® Itanium® 2 Processor Datasheet* | 253795 |
| *Intel® Itanium® 2 Processor Hardware Developer's Manual* | 251109 |
| *Intel® Itanium® Architecture Software Developer's Manual*, Volume 1: Application Architecture | 245317-005 |
| *Intel® Itanium® Architecture Software Developer's Manual*, Volume 2: System Architecture | 245318-005 |
| *Intel® Itanium® Architecture Software Developer's Manual*, Volume 3: Instruction Set Reference | 245319-005 |
| *Intel® Itanium® Architecture Software Developer's Manual Specification Update* | 248699 |
| *Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization* | 251110 |
| *Intel® Itanium® Processor Family System Abstraction Layer Specification* | 245359 |
| *Dual-Core Intel® Itanium® 2 Processor 9000 Series Datasheet* | 314054-001 |

## 2.2 Nomenclature

**S-Spec Number** is used to identify products. Products are differentiated by their unique characteristics, for example, core speed, L3 cache size, package types, and so forth. Care should be taken to read all notes associated with each S-Spec number.

**Errata** are design defects or errors. These may cause the Intel® Itanium® 2 processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given stepping must assume that all errata documented for that stepping are present on all devices.

**Specification Changes** are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

**Specification Clarifications** describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specification.

**Documentation Changes** include typos, errors, or omissions from the current published specifications. These changes are incorporated in the next release of the specifications.

***Note:*** Errata remain in the specification update throughout the product's life cycle or until a particular stepping is no longer commercially available. Under these circumstances, errata removed are archived and available upon request. Specification changes, specification clarifications, and documentation changes are removed when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, and so forth).



# 3 Summary Table of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the Intel Itanium 2 processors. Intel may fix some of the errata in a future stepping of the component or in a future release of the Processor Abstraction Layer (PAL), and account for the other outstanding issues through documentation or specification changes as noted. This table uses the notations indicated below.

## 3.1 Codes Used in Summary Table

### 3.1.1 Stepping/Version

X: Errata exists in the indicated stepping, PAL version, or software extension. Documentation Change, Specification Change or Clarification that applies to this stepping.

(No mark or Blank box): This erratum is fixed in listed stepping or specification change does not apply to listed stepping or PAL version.

### 3.1.2 Page

(Page): Page location of item in this document.

### 3.1.3 Status

Doc: Document change or update will be implemented.

Plan Fix: This erratum may be fixed in a future stepping of the component, or in a future release of PAL.

Fixed: This erratum has been previously fixed.

No Fix: There are no plans to fix this erratum.

### 3.1.4 Row

Change bar to left of table row indicates this erratum is either new or modified from the previous version of this document.

## Table 3-1. Definition Table

| Processor | Abbreviation |
|---|---|
| Intel® Itanium® 2 Processor 900 MHz with 1.5 MB L3 Cache | Itanium 2 Processor (up to 3 MB L3 cache) |
| Intel® Itanium® 2 Processor 1.0 GHz with 3 MB L3 Cache | |
| Low Voltage Intel® Itanium® 2 Processor 1.0 GHz with 1.5 MB L3 Cache | Itanium 2 Processor (up to 6 MB L3 cache) |
| Intel® Itanium® 2 Processor 1.40 GHz with 1.5 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.30 GHz with 3 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.40 GHz with 3 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.60 GHz with 3 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.40 GHz with 4 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.50 GHz with 6 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.50 GHz with 4 MB L3 Cache | Itanium 2 Processor (up to 9 MB L3 cache) |
| Intel® Itanium® 2 Processor 1.60 GHz with 6 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.60 GHz with 9 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.66 GHz with 6 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.66 GHz with 9 MB L3 Cache | |
| Low Voltage Intel® Itanium® 2 Processor 1.30 GHz with 3 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.60 GHz with 3 MB L3 Cache at 400 and 533 MHz System Bus (DP Optimized) | |
| Dual-Core Intel® Itanium® 2 Processor 1.6 GHz with 24 MB L3 Cache | Dual-Core Intel® Itanium® 2 Processor 9000 Series |
| Dual-Core Intel® Itanium® 2 Processor 1.6 GHz with 18 MB L3 Cache | |
| Dual-Core Intel® Itanium® 2 Processor 1.6 GHz with 8 MB L3 Cache | |
| Dual-Core Intel® Itanium® 2 Processor 1.42 GHz with 12 MB L3 Cache | |
| Dual-Core Intel® Itanium® 2 Processor 1.4 GHz with 12 MB L3 Cache | |
| Intel® Itanium® 2 Processor 1.6 GHz with 6 MB L3 Cache | |

## 3.2 Itanium® 2 Processor (up to 3 MB L3 Cache) Errata (Sheet 1 of 6)

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | | |
| 1 | X | | | | | | | | | | | | 35 | No Fix | IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED does not count predicated off instructions |
| 2 | X | | | | | | | | | | | | 35 | No Fix | Performance Monitor Interrupt raised when freeze bit is written to Performance Monitoring Counter register |
| 3 | X | | | | | | | | | | | | 35 | No Fix | Priority agent requests with unit mask of I/O not counted |
| 4 | X | | | | | | | | | | | | 35 | No Fix | Incorrect fault reporting on move to/from the RNAT or BSPSTORE application registers |
| 5 | X | | | | | | | | | | | | 36 | No Fix | Power good deassertion affects boundary scan testing |
| 6 | X | | | | | | | | | | | | 36 | No Fix | IA-32: CPUID instruction returns incorrect L3 cache size |
| 7 | X | | | | | | | | | | | | 36 | No Fix | Performance Monitoring Event counters may be incorrect when using Instruction Address Range checking in fine mode |
| 8 | X | | | | | | | | | | | | 36 | No Fix | Possible deadlock condition after ptc.g is issued on two-way system |
| 9 | X | | | | | | | | | | | | 37 | No Fix | EPC, mov ar.pfs and br.ret instructions may combine to yield incorrect privilege level |
| 10 | X | | | | | | | | | | | | 37 | No Fix | Removal of WAW hazard may lead to undefined result |
| 11 | X | | | | | | | | | | | | 38 | No Fix | Unexpected data debug, data access or dirty bit fault taken after rfi instruction |
| 12 | X | | | | | | | | | | | | 38 | No Fix | Incorrect privilege level may be granted if a failed speculation check precedes a privilege level change |
| 13 | X | | | | | | | | | | | | 38 | No Fix | Floating-point instructions take a floating-point trap before Unimplemented Instruction Address trap |
| 14 | | X | | | | | | | | | | | 39 | Fixed | PAL_MC_ERROR_INFO does not return an address for certain double bit ECC memory errors |
| 15 | | X | | | | | | | | | | | 39 | Fixed | PAL_CACHE_READ and PAL_CACHE_WRITE return incorrect status for L1I cache access |

## 3.2 Itanium® 2 Processor (up to 3 MB L3 Cache) Errata (Sheet 2 of 6)

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | | |
| 16 | | X | | | | | | | | | | | 39 | Fixed | Unpredictable behavior if the system is awakened from low power mode by an MCA |
| 17 | | X | | | | | | | | | | | 39 | Fixed | The system may lose an interrupt when SAL_CHECK reads the IVR |
| 18 | | X | | | | | | | | | | | 40 | Fixed | A bus MCA nested within a recoverable or firmware-corrected bus MCA may not be handled correctly |
| 19 | | X | | | | | | | | | | | 40 | Fixed | PAL reset sequence performed after a recovery check may result in incorrect system behavior |
| 20 | | X | | | | | | | | | | | 40 | Fixed | PAL_HALT_LIGHT_SPECIAL provides PAL_HALT functionality |
| 21 | | X | | | | | | | | | | | 40 | Fixed | PAL_TEST_PROC may access memory with the UC attribute |
| 22 | X | | | | | | | | | | | | 40 | No Fix | L2 single bit data error promoted to MCA continues to flag a CMCI |
| 23 | | X | | | | | | | | | | | 41 | Fixed | PAL_TEST_PROC requires specific tests be performed for correct operation |
| 24 | | X | | | | | | | | | | | 41 | Fixed | PAL_TEST_INFO may return incorrect data for invalid test parameters |
| 25 | | X | | | | | | | | | | | 41 | Fixed | PAL_CACHE_INIT may not function properly if levels of the cache hierarchy are specified |
| 26 | | X | | | | | | | | | | | 41 | Fixed | PAL_SET_TIMEOUT may have an unexpected result when time-out = 0 |
| 27 | | X | | | | | | | | | | | 41 | Fixed | Concurrent MCAs that signal a BERR may not set PSP.bc correctly |
| 28 | | X | | | | | | | | | | | 41 | Fixed | PAL_PLATFORM_ADDR may return an error if bit 63 is set |
| 29 | | X | | | | | | | | | | | 42 | Fixed | PAL_TEST_PROC may overwrite predicate registers |
| 30 | | X | | | | | | | | | | | 42 | Fixed | Recovery check fails if PAL_B is not found |
| 31 | | X | | | | | | | | | | | 42 | Fixed | PAL procedure calls may have unexpected results if an incorrect PAL_B version is used |
| 32 | | X | | | | | | | | | | | 42 | Fixed | Late self-test may have unexpected results during concurrent processor tests |
| 33 | | X | | | | | | | | | | | 42 | Fixed | PAL_TEST_PROC may cause unexpected system behavior |

## 3.2 Itanium® 2 Processor (up to 3 MB L3 Cache) Errata (Sheet 3 of 6)

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | | |
| 34 | | X | | | | | | | | | | | 43 | Fixed | PAL halt procedures may overwrite predicate registers |
| 35 | X | | | | | | | | | | | | 43 | No Fix | Two resets may be necessary to leave TAP test mode |
| 36 | X | | | | | | | | | | | | 43 | No Fix | IA-32 instruction pointers may be overwritten under certain boundary conditions |
| 37 | | X | | | | | | | | | | | 43 | Fixed | Initialization and ETM recovery may overwrite branch register |
| 38 | | | X | | | | | | | | | | 43 | Fixed | PAL procedures may not save predicate register 3 |
| 39 | | X | X | | | | | | | | | | 44 | Fixed | PAL_CACHE_INFO procedure may return undefined value |
| 40 | | | X | | | | | | | | | | 44 | Fixed | PAL_HALT_LIGHT procedure may generate a spurious Performance Monitor Interrupt |
| 41 | | X | X | | | | | | | | | | 44 | Fixed | Unexpected system behavior after PAL_CACHE_FLUSH is executed |
| 42 | | X | X | | | | | | | | | | 44 | Fixed | PAL_TEST_PROC may not properly report self-test status |
| 43 | X | | | | | | | | | | | | 44 | No Fix | PSR.ri may not reflect the correct slot upon entrance to the unimplemented address fault handler |
| 44 | X | | | | | | | | | | | | 45 | No Fix | WC and WB memory attribute aliasing combine with FC and may cause processor live-lock |
| 45 | X | | | | | | | | | | | | 45 | No Fix | Improper use of memory attribute aliasing may lead to out of order instruction execution |
| 47 | X | | | | | | | | | | | | 46 | No Fix | Executing an rfi instruction that is located at the end of implemented physical memory can result in an unexpected unimplemented address fault |
| 48 | X | | | | | | | | | | | | 46 | Fixed | IA-32: xchg instruction requires release semantics |
| 49 | | X | X | X | | | | | | | | | 46 | Fixed | PAL MCA handler may not correctly set PSP.co bit |
| 50 | | X | X | X | | | | | | | | | 46 | Fixed | PAL_MC_ERROR_INFO may return incorrect PSP information |
| 51 | X | | | | | | | | | | | | 47 | No Fix | FPSWA trap may be missed |

## 3.2 Itanium® 2 Processor (up to 3 MB L3 Cache) Errata (Sheet 4 of 6)

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | | |
| 52 | X | | | | | | | | | | | | 48 | Fixed | WC evictions and semaphore operations combine to establish a potential live-lock condition |
| 53 | X | | | | | | | | | | | | 48 | Fixed | The IA-32 cmpxchg8b instruction may not correctly set ZF flag |
| 54 | | X | X | X | X | X | X | X | X | X | X | X | 48 | No Fix | PAL_TEST_PROC status return value |
| 55 | X | | | | | | | | | | | | 49 | No Fix | Fault condition may generate incorrect address when using short format VHPT |
| 57 | | X | X | X | X | X | | | | | | | 49 | Fixed | Cache snoops disabled on BINIT# |
| 58 | X | | | | | | | | | | | | 50 | No Fix | RFI to UIA using single step mode may enter ss trap |
| 60 | X | | | | | | | | | | | | 50 | No Fix | Specific instruction combination may disrupt subsequent operation |
| 61 | X | | | | | | | | | | | | 50 | No Fix | IFS register may be invalidated during MCA or INIT |
| 62 | | | | | | | X | | | | | | 51 | Fixed | Unimplemented memory access may occur while handling an INIT or MCA event |
| 66 | | X | X | X | X | X | X | X | X | X | X | X | 52 | No Fix | PSP.cr is always set to zero (0) at PALE_INIT hand off to SALE_ENTRY |
| 68 | | X | X | X | X | X | X | | | | | | 53 | Fixed | Performance Monitoring Event counters may be incorrect after leaving a low-power state |
| 69 | X | | | | | | | | | | | | 53 | No Fix | Instruction Breakpoint Register update may generate a false instruction debug fault |
| 70 | X | | | | | | | | | | | | 53 | No Fix | Application fault may be missed on a br.ia instruction |
| 71 | X | | | | | | | | | | | | 53 | No Fix | Machine check may not bring the system out of a low-power state |
| 72 | | X | X | X | X | X | X | | | | | | 53 | Fixed | Machine check event received during PAL execution may have unexpected results |
| 73 | | X | X | X | X | X | X | | | | | | 54 | Fixed | Rendezvous may result in spin loop due to incorrect rendezvous address passed to SAL |
| 74 | | X | X | X | X | X | X | | | | | | 54 | Fixed | Possible degradation in system performance when calling PAL_CACHE_FLUSH with int = 1 for certain cache memory types |

## 3.2 Itanium® 2 Processor (up to 3 MB L3 Cache) Errata (Sheet 5 of 6)

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | | |
| 75 | X | | | | | | | | | | | | 54 | No Fix | Memory read current transaction may fail to observe a st, ld.bias or lfetch.excl |
| 76 | X | | | | | | | | | | | | 55 | No Fix | BINIT taken on 2x ECC and hard-fail errors with BINIT event signaling disabled |
| 77 | X | | | | | | | | | | | | 55 | No Fix | Recoverable L3 cache tag ECC error may raise overflow error when CMCI are promoted to MCA |
| 78 | X | | | | | | | | | | | | 55 | No Fix | L2 cache line with poison data results in unexpected fatal MCA |
| 79 | X | | | | | | | | | | | | 55 | No Fix | XPN time-out with BINIT response disabled may cause system hang |
| 80 | X | | | | | | | | | | | | 56 | No Fix | BINIT may be taken after a UC single byte access to ignored/reserved area of the Processor Interrupt Block |
| 81 | X | | | | | | | | | | | | 56 | No Fix | Recoverable CMCI may combine with an L3 MCA error to cause fatal overflow error |
| 82 | | X | X | X | X | X | X | | | | | | 56 | Fixed | BERR may be indicated when the PAL MCA routine invalidates L2 cache lines |
| 83 | | X | X | X | X | X | X | X | | | | | 56 | Fixed | Pending RSE interrupt during the PAL PMI handler PAL PMI flow may result in a system hang |
| 84 | | X | X | X | X | X | X | X | X | X | X | X | 56 | No Fix | An INIT signaled during the PAL PMI flow while a PAL PMI flow RFI is being serviced may result in a system hang |
| 85 | | X | X | X | X | X | X | X | | | | | 57 | Fixed | PMI serviced during the execution of PAL_MCMA_ERROR_INFO procedure may result in unpredictable processor behavior |
| 86 | | X | X | X | X | X | X | X | X | X | X | X | 57 | No Fix | Data-poisoning bits not included in PAL_MC_ERROR_INFO cache_check and bus_check structures |
| 87 | | X | X | X | X | X | X | X | X | | | | 57 | Fixed | PAL_PREFETCH_VISIBILITY call not implemented |
| 89 | | X | X | X | X | X | X | X | X | | | | 58 | Fixed | Cache lines with ECC errors may not be invalidated |
| 90 | | X | X | X | X | X | X | X | X | | | | 58 | Fixed | Interrupts are enabled when exiting from a halt state |
| 92 | X | | | | | | | | | | | | 58 | No Fix | Corrected ECC error may not generate CMCI |

## 3.2 Itanium® 2 Processor (up to 3 MB L3 Cache) Errata (Sheet 6 of 6)

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | | |
| 93 | | X | X | X | X | X | X | X | X | | | | 58 | Fixed | PAL_CACHE_FLUSH procedure may not flush and invalidate all L2 cache lines |
| 94 | | X | X | X | X | X | X | X | X | | | | 59 | Fixed | Performance counters may include data from low power states |
| 95 | | X | X | X | X | X | X | X | X | | | | 59 | Fixed | MCA due to an XPN timeout may result in a spin loop |
| 96 | X | | | | | | | | | | | | 59 | No Fix | BINIT# may not be asserted for exactly two cycles |
| 97 | X | | | | | | | | | | | | 60 | No Fix | Memory read current transaction may fail to observe a st or lead to a system hang |
| 98 | | X | X | X | X | X | X | X | X | X | | | 60 | Fixed | PAL_VM_TR_READ will return an incorrect page size for DTR reads |
| 100 | | X | X | X | X | X | X | X | X | X | | | 60 | Fixed | Interruption of PAL calls by a PMI or INIT |
| 102 | | X | X | X | X | X | X | X | X | X | | | 61 | Fixed | PAL_MC_ERROR_INFO call could invalidate incorrect cache line entry |
| 104 | | X | X | X | X | X | X | X | X | X | | | 61 | Fixed | SALE_ENTRY may see unexpected modified cache line during system cold boot |
| 105 | X | | | | | | | | | | | | 61 | No Fix | Lower priority error flagged on illegal write to GR r0 |
| 107 | | X | X | X | X | X | X | X | X | X | X | X | 62 | No Fix | PAL_CAR_INIT may not clear all cache lines |
| 108 | | | | | | | | | | | X | | 62 | Fixed | PSR.IC may not be restored properly on exit from a PAL call |
| 109 | | X | X | X | X | X | X | X | X | X | X | | 62 | Fixed | Performance counters may not be correctly restored upon exit of the LIGHT HALT state |
| 110 | X | X | X | X | X | X | X | X | X | X | X | X | 63 | Plan Fix | Single-bit errors in the tag and data portion of cache lines in the “I” state in the L2 or L3 levels of cache may not be flushed |
| 111 | X | X | X | X | X | X | X | X | X | X | X | X | 63 | No Fix | Un-initialized word lines at processor boot could result in an incorrect branch address |
| 155 | | | | | | | | | | | X | X | 71 | No Fix | Processors may not wake from the LIGHT HALT state upon MCA |
| 175 | | X | X | X | X | X | X | X | X | X | X | X | 75 | Plan Fix | Poison data in the caches has partial or no indication of 2xECC error when written back to memory |
| 184 | | X | X | X | X | X | X | X | X | X | X | X | 76 | Plan Fix | Calls to PAL_MC_ERROR_INFO could cause a processor hang |

## 3.3 Itanium® 2 Processor (up to 6 MB L3 Cache) Errata (Sheet 1 of 3)

| No. | Processor Stepping | PAL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|
| | B1 | 5.37 | 5.61 | 5.65 | 5.69 | 5.72 | 5.73 | | | |
| 1 | X | | | | | | | 35 | No Fix | IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED does not count predicated off instructions |
| 2 | X | | | | | | | 35 | No Fix | Performance Monitor Interrupt raised when freeze bit is written to Performance Monitoring Counter register |
| 6 | X | | | | | | | 36 | No Fix | IA-32: CPUID instruction returns incorrect L3 cache size |
| 7 | X | | | | | | | 36 | No Fix | Performance Monitoring Event counters may be incorrect when using Instruction Address Range checking in fine mode |
| 8 | X | | | | | | | 36 | No Fix | Possible deadlock condition after ptc.g is issued on two-way system |
| 13 | X | | | | | | | 38 | No Fix | Floating-point instructions take a floating-point trap before Unimplemented Instruction Address trap |
| 22 | X | | | | | | | 40 | No Fix | L2 single bit data error promoted to MCA continues to flag a CMCI |
| 43 | X | | | | | | | 44 | No Fix | PSR.ri may not reflect the correct slot upon entrance to the unimplemented address fault handler |
| 45 | X | | | | | | | 45 | No Fix | Improper use of memory attribute aliasing may lead to out of order instruction execution |
| 47 | X | | | | | | | 46 | No Fix | Executing an rfi instruction that is located at the end of implemented physical memory can result in an unexpected unimplemented address fault |
| 54 | | X | X | X | X | X | X | 48 | No Fix | PAL_TEST_PROC status return value |
| 55 | X | | | | | | | 49 | No Fix | Fault condition may generate incorrect address when using short format VHPT |
| 58 | X | | | | | | | 50 | No Fix | RFI to UIA using single step mode may enter ss trap |
| 59 | X | | | | | | | 50 | No Fix | On-Die Termination value does not meet specification |
| 61 | X | | | | | | | 50 | No Fix | IFS register may be invalidated during MCA or INIT |
| 62 | | X | | | | | | 51 | Fixed | Unimplemented memory access may occur while handling an INIT or MCA event |
| 63 | X | | | | | | | 51 | No Fix | JTAG Sample/Preload or EXTEST instruction usage |
| 64 | X | | | | | | | 52 | Fixed | CPU_CYCLES count includes data from halt states |
| 65 | X | | | | | | | 52 | No Fix | System bus signals can be driven while RESET# is asserted |
| 66 | | X | X | X | X | X | X | 52 | No Fix | PSP.cr is always set to zero (0) at PALE_INIT hand off to SALE_ENTRY |
| 67 | X | | | | | | | 52 | No Fix | Incorrect Thermal Calibration Offset Byte value in the PIROM |
| 69 | X | | | | | | | 53 | No Fix | Instruction Breakpoint Register update may generate a false instruction debug fault |
| 70 | X | | | | | | | 53 | No Fix | Application fault may be missed on a br.ia instruction |
| 71 | X | | | | | | | 53 | No Fix | Machine check may not bring the system out of a low-power state |
| 72 | | X | | | | | | 53 | Fixed | Machine check event received during PAL execution may have unexpected results |
| 73 | | X | | | | | | 54 | Fixed | Rendezvous may result in spin loop due to incorrect rendezvous address passed to SAL |
| 74 | | X | | | | | | 54 | Fixed | Possible degradation in system performance when calling PAL_CACHE_FLUSH with int = 1 for certain cache memory types |
| 75 | X | | | | | | | 54 | No Fix | Memory read current transaction may fail to observe a st, ld.bias or lfetch.excl |
| 76 | X | | | | | | | 55 | No Fix | BINIT taken on 2x ECC and hard-fail errors with BINIT event signaling disabled |
| 77 | X | | | | | | | 55 | No Fix | Recoverable L3 cache tag ECC error may raise overflow error when CMCI are promoted to MCA |

## 3.3 Itanium® 2 Processor (up to 6 MB L3 Cache) Errata (Sheet 2 of 3)

| No. | Processor Stepping | PAL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|
| | B1 | 5.37 | 5.61 | 5.65 | 5.69 | 5.72 | 5.73 | | | |
| 78 | X | | | | | | | 55 | No Fix | L2 cache line with poison data results in unexpected fatal MCA |
| 79 | X | | | | | | | 55 | No Fix | XPN time-out with BINIT response disabled may cause system hang |
| 80 | X | | | | | | | 56 | No Fix | BINIT may be taken after a UC single byte access to ignored/reserved area of the Processor Interrupt Block |
| 81 | X | | | | | | | 56 | No Fix | Recoverable CMCI may combine with an L3 MCA error to cause fatal overflow error |
| 82 | | X | | | | | | 56 | Fixed | BERR may be indicated when the PAL MCA routine invalidates L2 cache lines |
| 83 | | X | X | | | | | 56 | Fixed | Pending RSE interrupt during the PAL PMI handler PAL PMI flow may result in a system hang |
| 84 | | X | X | X | X | X | X | 56 | No Fix | An INIT signaled during the PAL PMI flow while a PAL PMI flow RFI is being serviced may result in a system hang |
| 85 | | X | X | | | | | 57 | Fixed | PMI serviced during the execution of PAL_MCMA_ERROR_INFO procedure may result in unpredictable processor behavior |
| 86 | | X | X | | | X | X | 57 | No Fix | Data-poisoning bits not included in PAL_MC_ERROR_INFO cache_check and bus_check structures |
| 87 | | | X | | | | | 57 | Fixed | PAL_PREFETCH_VISIBILITY call not implemented |
| 88 | X | | | | | | | 57 | No Fix | INIT# signal not recognized properly |
| 89 | | X | X | X | | | | 58 | Fixed | Cache lines with ECC errors may not be invalidated |
| 90 | | X | X | X | | | | 58 | Fixed | Interrupts are enabled when exiting from a halt state |
| 91 | | | | X | | | | 58 | Fixed | PAL_PREFETCH_VISIBILITY call may result in a system hang |
| 92 | X | | | | | | | 58 | No Fix | Corrected ECC error may not generate CMCI |
| 93 | | X | X | X | | | | 58 | Fixed | PAL_CACHE_FLUSH procedure may not flush and invalidate all L2 cache lines |
| 94 | | X | X | X | | | | 59 | Fixed | Performance counters may include data from low power states |
| 95 | | X | X | X | | | | 59 | Fixed | MCA due to an XPN timeout may result in a spin loop |
| 96 | X | | | | | | | 59 | No Fix | BINIT# may not be asserted for exactly two cycles |
| 97 | X | | | | | | | 60 | No Fix | Memory read current transaction may fail to observe a st or lead to a system hang |
| 98 | | X | X | X | X | | | 60 | Fixed | PAL_VM_TR_READ will return an incorrect page size for DTR reads |
| 100 | | X | X | X | X | | | 60 | Fixed | Interruption of PAL calls by a PMI or INIT |
| 102 | | X | X | X | X | | | 61 | Fixed | PAL_MC_ERROR_INFO call could invalidate incorrect cache line entry |
| 104 | | X | X | X | X | | | 61 | Fixed | SALE_ENTRY may see unexpected modified cache line during system cold boot |
| 105 | X | | | | | | | 61 | No Fix | Lower priority error flagged on illegal write to GR r0 |
| 106 | | X | X | X | X | | | 62 | Fixed | PAL_TEST_PROC L3 cache replacement test may return invalid response |
| 107 | | | X | X | X | X | X | 62 | No Fix | PAL_CAR_INIT may not clear all cache lines |
| 108 | | | | | | X | | 62 | Fixed | PSR.IC may not be restored properly on exit from a PAL call |
| 109 | | X | X | X | X | X | | 62 | Fixed | Performance counters may not be correctly restored upon exit of the LIGHT HALT state |
| 110 | X | X | X | X | X | X | X | 63 | Plan Fix | Single-bit errors in the tag and data portion of cache lines in the “I” state in the L2 or L3 levels of cache may not be flushed |
| 111 | X | X | X | X | X | X | X | 63 | No Fix | Un-initialized word lines at processor boot could result in an incorrect branch address |

## 3.3 Itanium® 2 Processor (up to 6 MB L3 Cache) Errata (Sheet 3 of 3)

| No. | Processor Stepping | PAL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|
| | B1 | 5.37 | 5.61 | 5.65 | 5.69 | 5.72 | 5.73 | | | |
| 155 | | | | | | X | X | 71 | No Fix | Processors may not wake from the LIGHT HALT state upon MCA |
| 175 | | X | X | X | X | X | X | 75 | Plan Fix | Poison data in the caches has partial or no indication of 2xECC error when written back to memory |
| 184 | | X | X | X | X | X | X | 76 | Plan Fix | Calls to PAL_MC_ERROR_INFO could cause a processor hang |

## 3.4 Itanium® 2 Processor (up to 9 MB L3 Cache) Errata (Sheet 1 of 2)

| No. | Processor Stepping | | PAL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | A1 | A2 | 1.27 | 2.10 | 2.14 | 2.15 | 2.20 | 2.24 | | | |
| 1 | X | X | | | | | | | 35 | No Fix | IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED does not count predicated off instructions |
| 2 | X | X | | | | | | | 35 | No Fix | Performance Monitor Interrupt raised when freeze bit is written to Performance Monitoring Counter register |
| 6 | X | X | | | | | | | 36 | No Fix | IA-32: CPUID instruction returns incorrect L3 cache size |
| 7 | X | X | | | | | | | 36 | No Fix | Performance Monitoring Event counters may be incorrect when using Instruction Address Range checking in fine mode |
| 8 | X | X | | | | | | | 36 | No Fix | Possible deadlock condition after ptc.g is issued on two-way system |
| 13 | X | X | | | | | | | 38 | No Fix | Floating-point instructions take a floating-point trap before Unimplemented Instruction Address trap |
| 22 | X | X | | | | | | | 40 | No Fix | L2 single bit data error promoted to MCA continues to flag a CMCI |
| 43 | X | X | | | | | | | 44 | No Fix | PSR.ri may not reflect the correct slot upon entrance to the unimplemented address fault handler |
| 45 | X | X | | | | | | | 45 | No Fix | Improper use of memory attribute aliasing may lead to out of order instruction execution |
| 47 | X | X | | | | | | | 46 | No Fix | Executing an rfi instruction that is located at the end of implemented physical memory can result in an unexpected unimplemented address fault |
| 54 | | X | X | X | X | X | | | 48 | No Fix | PAL_TEST_PROC status return value |
| 55 | X | X | | | | | | | 49 | No Fix | Fault condition may generate incorrect address when using short format VHPT |
| 58 | X | X | | | | | | | 50 | No Fix | RFI to UIA using single step mode may enter ss trap |
| 63 | X | X | | | | | | | 51 | No Fix | JTAG Sample/Preload or EXTEST instruction usage |
| 66 | | X | X | X | X | X | | | 52 | No Fix | PSP.cr is always set to zero (0) at PALE_INIT hand off to SALE_ENTRY |
| 67 | X | X | | | | | | | 52 | No Fix | Incorrect Thermal Calibration Offset Byte value in the PIROM |
| 69 | X | X | | | | | | | 53 | No Fix | Instruction Breakpoint Register update may generate a false instruction debug fault |
| 70 | X | X | | | | | | | 53 | No Fix | Application fault may be missed on a br.ia instruction |

## 3.4 Itanium® 2 Processor (up to 9 MB L3 Cache) Errata (Sheet 2 of 2)

| No. | Processor Stepping | | PAL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | A1 | A2 | 1.27 | 2.10 | 2.14 | 2.15 | 2.20 | 2.24 | | | |
| 71 | X | X | | | | | | | 53 | No Fix | Machine check may not bring the system out of a low-power state |
| 76 | X | X | | | | | | | 55 | No Fix | BINIT taken on 2x ECC and hard-fail errors with BINIT event signaling disabled |
| 77 | X | X | | | | | | | 55 | No Fix | Recoverable L3 cache tag ECC error may raise overflow error when CMCI are promoted to MCA |
| 79 | X | X | | | | | | | 55 | No Fix | XPN time-out with BINIT response disabled may cause system hang |
| 80 | X | X | | | | | | | 56 | No Fix | BINIT may be taken after a UC single byte access to ignored/reserved area of the Processor Interrupt Block |
| 84 | | | X | X | X | X | | | 56 | No Fix | An INIT signaled during the PAL PMI flow while a PAL PMI flow RFI is being serviced may result in a system hang |
| 86 | | | X | X | X | X | | | 57 | No Fix | Data-poisoning bits not included in PAL_MC_ERROR_INFO cache_check and bus_check structures |
| 96 | X | X | | | | | | | 59 | No Fix | BINIT# may not be asserted for exactly two cycles |
| 97 | X | | | | | | | | 60 | Fixed | Memory read current transaction may fail to observe a st or lead to a system hang |
| 98 | | | X | | | | | | 60 | Fixed | PAL_VM_TR_READ will return an incorrect page size for DTR reads |
| 99 | | | X | | | | | | 60 | Fixed | Incorrect EID and ID information passed by PAL |
| 100 | | | X | | | | | | 60 | Fixed | Interruption of PAL calls by a PMI or INIT |
| 101 | | | X | | | | | | 61 | Fixed | External interrupt polling and PAL_CACHE_FLUSH |
| 102 | | | X | | | | | | 61 | Fixed | PAL_MC_ERROR_INFO call could invalidate incorrect cache line entry |
| 103 | X | | | | | | | | 61 | Fixed | L3 cache tag error and pending cache line replacement transactions may result in system livelock |
| 104 | | | X | | | | | | 61 | Fixed | SALE_ENTRY may see unexpected modified cache line during system cold boot |
| 105 | X | X | | | | | | | 61 | No Fix | Lower priority error flagged on illegal write to GR r0 |
| 107 | | | X | X | X | X | | | 62 | No Fix | PAL_CAR_INIT may not clear all cache lines |
| 108 | | | | X | | | | | 62 | Fixed | PSR.IC may not be restored properly on exit from a PAL call |
| 109 | | | X | X | | | | | 62 | Fixed | Performance counters may not be correctly restored upon exit of the LIGHT HALT state |
| 110 | X | X | X | X | X | X | | | 63 | Fixed | Single-bit errors in the tag and data portion of cache lines in the “I” state in the L2 or L3 levels of cache may not be flushed |
| 111 | X | X | X | X | X | X | | | 63 | No Fix | Un-initialized word lines at processor boot could result in an incorrect branch address |
| 155 | | | X | X | X | X | X | | 71 | No Fix | Processors may not wake from the LIGHT HALT state upon MCA |
| 167 | X | X[1] | | | | | | | 73 | Fixed | Potential electrical marginality in the integer register file |
| 175 | | | X | X | X | X | X | | 75 | Fixed | Poison data in the caches has partial or no indication of 2xECC error when written back to memory |
| 184 | | | X | X | X | X | X | | 76 | Plan Fix | Calls to PAL_MC_ERROR_INFO could cause a processor hang |

1. Fixed in Madison 9M processors shipped after September 2006 identified with an “A2” designation on the pin side of the processor.

## 3.5 Dual-Core Intel® Itanium® 2 Processor 9000 Series Errata (Sheet 1 of 3)

| No. | Processor Stepping | | PAL Version | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|
| | C1 | C2 | 7.46 | 8.30 | | | |
| 1 | X | | | | 35 | No Fix | IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED does not count predicated off instructions |
| 8 | X | | | | 36 | No Fix | Possible deadlock condition after ptc.g is issued on two-way system |
| 13 | X | | | | 38 | No Fix | Floating-point instructions take a floating-point trap before Unimplemented Instruction Address trap |
| 45 | X | | | | 45 | No Fix | Improper use of memory attribute aliasing may lead to out of order instruction execution |
| 47 | X | | | | 46 | No Fix | Executing an rfi instruction that is located at the end of implemented physical memory can result in an unexpected unimplemented address fault |
| 54 | X | | | | 48 | No Fix | PAL_TEST_PROC status return value |
| 63 | X | | | | 51 | No Fix | JTAG Sample/Preload or EXTEST instruction usage |
| 66 | X | | | | 52 | No Fix | PSP.cr is always set to zero (0) at PALE_INIT hand off to SALE_ENTRY |
| 67 | X | | | | 52 | No Fix | Incorrect Thermal Calibration Offset Byte value in the PIROM |
| 70 | X | | | | 53 | No Fix | Application fault may be missed on a br.ia instruction |
| 79 | X | | | | 55 | No Fix | XPN time-out with BINIT response disabled may cause system hang |
| 80 | X | | | | 56 | No Fix | BINIT may be taken after a UC single byte access to ignored/reserved area of the Processor Interrupt Block |
| 86 | X | | | | 57 | Fixed | Data-poisoning bits not included in PAL_MC_ERROR_INFO cache_check and bus_check structures |
| 112 | X | | | | 63 | No Fix | Unexpected MCA on a fill to a line with parity errors |
| 113 | X | | | | 63 | No Fix | Performance associated with an epc instruction |
| 114 | X | | | | 63 | No Fix | Branch bit, mispredict bit and slot index of branch instruction |
| 115 | X | | | | 64 | No Fix | Lower priority error flagged on illegal write to GR r0 |
| 116 | X | | | | 64 | No Fix | ptc.e instructions may purge resources of the other logical processor executing on the same core |
| 117 | X | | | | 64 | No Fix | MPE_SCB_LIVE_REQ counts for disabled cores |
| 118 | X | | | | 64 | No Fix | move to bspstore requires unexpected serialization |
| 119 | X | | | | 64 | No Fix | System behavior as a result of nested BINIT’s |
| 120 | X | | | | 65 | No Fix | Instruction Pointer-Event Address Register (IP-EAR) may not behave as specified |
| 121 | X | | | | 65 | No Fix | Performance Monitor Data (PMD) registers 10-15 usage |
| 122 | | | X | | 65 | No Fix | Wrong address generated for L3 data 1x and 2x ECC errors |
| 123 | | | X | | 65 | No Fix | Illegal opcodes may not raise the expected operation fault |
| 124 | | | X | | 65 | Fixed | Logical Processor Migration (LPM) is not working as expected |
| 125 | | | X | | 66 | Fixed | ALAT test is unavailable |
| 126 | | | X | | 66 | Fixed | Internal processor timeout (XPN) events are not signaled |
| 127 | | | X | | 66 | Fixed | PAL incorrectly interpreting updates to the Virtual Processor Descriptor |
| 128 | | | X | | 66 | Fixed | PAL based IA-32 execution may result in unpredictable behavior |
| 129 | X | X | | | 66 | No Fix | Two MCAs issued due to active logical processor being switched |
| 130 | | | X | | 66 | Fixed | PAL_VP_SAVE and PAL_VP_RESTORE procedures not working as expected |
| 131 | | | X | | 67 | Fixed | PAL_VP_REGISTER procedure not working as expected |

## 3.5 Dual-Core Intel® Itanium® 2 Processor 9000 Series Errata (Sheet 2 of 3)

| No. | Processor Stepping | | PAL Version | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|
| | C1 | C2 | 7.46 | 8.30 | | | |
| 132 | | | X | | 67 | Fixed | Write access to a cache line with an uncorrectable error results in a MCA instead of a CMCI |
| 133 | | | X | | 67 | Fixed | PAL_CACHE_SHARED_INFO not working as expected |
| 134 | | | X | | 67 | Fixed | PAL_MC_ERROR_INJECT in the cache error consumption mode may not work as expected |
| 135 | | | X | | 67 | Fixed | TLB consumption mode in PAL_MC_ERROR_INJECT uses the incorrect address |
| 136 | | | X | | 67 | Fixed | PAL correction of any L2D or L3 correctable error on a cache line may flush that line |
| 137 | | | X | | 68 | Fixed | PAL improperly decodes the instruction in response to a virtualization fault |
| 138 | | | X | | 68 | Fixed | BINIT# assertion may result in a system hang. |
| 139 | | | X | | 68 | Fixed | PAL is affecting ITP's ability to halt logical processors |
| 140 | | | X | | 68 | Fixed | PAL_MC_ERROR_INJECT not working as expected in the inject_only mode and the inject_and_consume mode |
| 141 | | | X | | 68 | Fixed | PAL incorrectly change the value for isr.code |
| 142 | | | X | | 69 | Fixed | Illegal operation faults of the type .lx are incorrectly delivered to the VMM virtualization fault vector |
| 143 | | | X | | 69 | Fixed | PAL incorrectly routes an illegal operation fault on a reset system mask(rsm) instruction fault |
| 144 | | | X | | 69 | Fixed | PAL based IA-32 execution does not raise single step trap |
| 145 | | | X | | 69 | Fixed | PAL based IA-32 execution does not respond to IA-32 debug traps |
| 146 | | | X | | 69 | Fixed | Intel® Cache Safe Technology "performance restricted" CMCI is issued after 3 ways per set are disabled. |
| 147 | | | X | | 69 | Fixed | Interval Time Counter (ITC) may not be properly initialized |
| 148 | | | X | | 70 | Plan Fix | POPF instruction may not be intercepted during PAL based IA-32 execution |
| 149 | | | X | | 70 | No Fix | CMCIs issued noting entry and exit from ETM even when ETM is disabled |
| 150 | | | X | | 70 | Fixed | Exclusion of first 3 single bit errors by Intel® Cache Safe Technology may cause system hangs in processors that have their L3 cache size equal to 6MB |
| 151 | | | X | | 70 | Fixed | Value of the IA-32 interruption code (ISR.code) is incorrectly set |
| 152 | X | | | | 70 | Fixed | Infinite snoop stall during RESET or BINIT |
| 153 | X | | | | 71 | Fixed | Clock misalignment may result in a loss of socket level lockstep |
| 154 | X | | | | 71 | Fixed | Execution of an instruction in the multi-media unit may result in unexpected behavior |
| 156 | | | X | | 71 | Fixed | Logical processor may be lost when a recoverable or a PAL-correctable MCA occurs during PAL_HALT_LIGHT |
| 157 | X | X | | | 71 | No Fix | On Die Termination (ODT) may be unexpectedly enabled |
| 158 | | | X | | 72 | Fixed | Failure to set the PSR.it bit to its original value |
| 159 | | | X | | 72 | Fixed | The PAL_PSTATE_INFO procedure may write to scratch floating point (FP) registers without saving and restoring the value of PSR.mfl |
| 160 | X | X | | | 72 | No Fix | Performance Monitor Unit (PMU) readings for system interface events may reflect both threads |
| 161 | | | X | | 72 | Plan Fix | PAL_FREQ_RATIO returns an incorrect value for 1.42 GHz parts |
| 162 | | | X | | 72 | Plan Fix | PAL_MC_CLEAR_LOG called on one logical processor may erase the processor error logs |
| 163 | | | X | | 73 | Fixed | Unable to specify the Current Frame Load Enable (CFLE) value at the target guest handler. |
| 164 | | | X | | 73 | No Fix | Infinite snoop stalls may be observed |
| 165 | | | X | | 73 | Fixed | Unexpected behavior when code request completes during PAL authentication |

## 3.5 Dual-Core Intel® Itanium® 2 Processor 9000 Series Errata (Sheet 3 of 3)

| No. | Processor Stepping | | PAL Version | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|
| | C1 | C2 | 7.46 | 8.30 | | | |
| 166 | | | X | | 73 | No Fix | PAL_CACHE_INFO is not available during firmware recovery check |
| 168 | | | X | X | 73 | Plan Fix | PAL_MC_ERROR_INJECT consume mode may not behave as expected |
| 169 | | | | X | 74 | Plan Fix | Using PAL_CONTEXT_RESTORE and PAL_CONTEXT_SAVE may result in a system hang during logical processor migration |
| 170 | | | X | X | 74 | Plan Fix | PAL_MC_ERROR_INFO may report an invalid index field |
| 171 | | | X | X | 74 | Plan Fix | PAL_BUS_SET_FEATURES bit 52 enables a bus cache line replacement transaction only when a cache line is in the shared state |
| 172 | | | X | X | 74 | Plan Fix | MOVL instructions taking a general exception fault are decoded as legal virtualized instructions |
| 173 | | | X | X | 74 | Plan Fix | Reserved register field fault checks do not check the present bit to determine if a reserved register field fault should be raised |
| 174 | | | X | X | 74 | No Fix | Calling PAL_CAR_INIT in cacheable mode may cause undefined behavior |
| 175 | X | X | | | 75 | No Fix | Poison data in the caches has partial or no indication of 2xECC error when written back to memory |
| 176 | | X | | X | 75 | No Fix | Multiple BINIT# assertions due to internal processor timeout (XPN) events |
| 177 | | | | X | 75 | Plan Fix | MSR_L0I_CONFIG corruption during PAL_CACHE_INIT, PAL_CAR_INIT, and reset |
| 178 | | | | X | 75 | Plan Fix | PAL_HALT_INFO returns an inaccurate value for power savings information |
| 179 | | | | X | 75 | Plan Fix | PAL_SET_HW_POLICY uses uninitialized register to initialize thread priority |
| 180 | | | | X | 76 | Plan Fix | PAL_SET_HW_POLICY may not preserve predicate bit p5 |
| 181 | | | | X | 76 | Plan Fix | PAL_MC_RESUME clears branch registers b6, b7 |
| 182 | X | X | | | 76 | No Fix | Snooped L3 tag and/or state ECC error sometimes reports wrong address |
| 183 | | | | X | 76 | Plan Fix | PAL_PSTATE_INFO returns data not compliant with the SDM |

## 3.6 FPSWA Errata

| No. | FPSWA Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|
| | 1.09 | 1.12 | 1.18 | | | | | | |
| 46 | X | | | | | | 45 | Fixed | FPSWA may not set the Denormal status flag correctly |
| 56 | | X | | | | | 49 | Fixed | FPSWA version 1.12 may overwrite register fr12 |

## 3.7 IA-32 Execution Layer Errata (Sheet 1 of 3)

| No. | IA-32 EL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|
| | 4.3 | 4.4 | 5.3 | 6.5 | | | | | |
| 1 | X | X | X | X | | | 80 | No Fix | Ordering of loads and stores |
| 2 | X | X | X | X | | | 80 | No Fix | Segmentation not supported |
| 3 | X | X | X | X | | | 80 | No Fix | 16-bit application mode not supported |
| 4 | X | X | X | X | | | 80 | No Fix | IA-32 floating-point state |
| 5 | X | X | X | X | | | 81 | No Fix | Floating-point C1 condition code flag support |
| 6 | X | X | X | X | | | 81 | No Fix | IA-32 floating-point pseudo-denormal, pseudo-NaN, and pseudo-infinity support |
| 7 | X | X | X | X | | | 81 | No Fix | Behavior of quiet and signaling NaNs |

## 3.7 IA-32 Execution Layer Errata (Sheet 2 of 3)

| No. | IA-32 EL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|
| | 4.3 | 4.4 | 5.3 | 6.5 | | | | | |
| 8 | X | X | X | X | | | 81 | No Fix | IA-32 floating-point exceptions |
| 9 | X | X | X | X | | | 81 | No Fix | Partial support for EFLAGS |
| 10 | X | X | X | X | | | 82 | No Fix | EFLAGS and floating-point exception flag behavior |
| 11 | X | X | X | X | | | 82 | No Fix | RSM and IRET instructions raise incorrect faults |
| 12 | X | X | X | X | | | 82 | No Fix | Cross-modifying code |
| 13 | X | X | X | X | | | 82 | No Fix | Atomicity of lock-prefixed instructions making unaligned memory references |
| 14 | X | X | X | X | | | 82 | No Fix | Atomicity of lock-prefixed instructions making uncacheable memory references |
| 15 | X | X | X | X | | | 83 | No Fix | Noninterruptability of 32-bit unaligned and 16-byte stores |
| 16 | X | | | | | | 83 | Fixed | IA-32 execution layer install and uninstall failures |
| 17 | X | | | | | | 83 | Fixed | Self-modifying code on unaligned memory may result in an access violation |
| 18 | X | X | X | | | | 83 | Fixed | Large data file accesses may return incorrect data |
| 19 | X | X | X | | | | 83 | Fixed | IA-32 EL applications will not run on kernels with page sizes greater than 16k |
| 20 | | | X | | | | 84 | Fixed | IA-32 EL may incorrectly optimize frequently executed code with interleaved integer and floating-point flag operations that include producer/consumer code sequences |
| 21 | X | X | X | | | | 84 | Fixed | IA-32 code running with the IA-32 EL may see an SSE Exception being ignored after the FPREM1 instruction is executed |
| 22 | | | X | | | | 84 | Fixed | An IA-32 EL optimized code procedure with interleaved MMX™ and SSE code may experience an application hang |
| 23 | | | X | | | | 85 | Fixed | An IA-32 Linux* application may receive an unexpected memory access violation |
| 24 | X | X | X | | | | 85 | Plan Fix | Wrong NEG EFlags cases |
| 25 | | | X | | | | 85 | Plan Fix | Lock XADD atomicity |
| 26 | | | X | | | | 85 | Plan Fix | Lock <***> + MOV weak order |
| 27 | | | X | | | | 86 | Plan Fix | SSE with behavior change |
| 28 | | | X | | | | 86 | Plan Fix | Thread not suspended |
| 29 | X | X | X | | | | 86 | Plan Fix | Extended-double to double precision |
| 30 | | | X | | | | 86 | Plan Fix | CMPXCHG EAX, reg |
| 31 | X | X | X | | | | 86 | Plan Fix | SSE with early loop exit |
| 32 | | | X | | | | 87 | Plan Fix | Exception/suspension in fnstsw-sahf-jcc |
| 33 | X | X | X | | | | 87 | Plan Fix | Load-misalign-reload |
| 34 | | | X | | | | 87 | Plan Fix | Incorrect register values in multi-block prefetch |
| 35 | | | X | | | | 88 | Plan Fix | Suspension while SMC observed |
| 36 | | | X | | | | 88 | Plan Fix | LINUX internal synchronization |
| 37 | X | X | X | | | | 88 | No Fix | Page crosser lock w/ permission change |
| 38 | X | X | X | | | | 88 | Fixed | Socketcall send/receive message may fail |
| 39 | X | X | X | | | | 89 | Fixed | Interrupted long Linux system call that receives an interruption-indication may unexpectedly modify an application buffer |
| 40 | | | X | | | | 89 | Fixed | ZF flag may be mishandled when using a CMPXCHG8b in an If-Then-Else code structure |
| 41 | | | | X | | | 89 | No Fix | Performing SSE divide of a denormal value by zero, while the DAZ bit is set, will result in a zero-divide exception instead of invalid-operation exception |
| 42 | X | X | X | X | | | 90 | No Fix | Asynchronous suspend and resume calls to a thread may result in undefined behavior |
| 43 | X | X | X | X | | | 90 | No Fix | Files under /proc/<pid> may contain incorrect data for emulated processes |
| 44 | X | X | X | X | | | 90 | No Fix | Select pending signals and SIG_IGN dispositions are not inherited cross-execve |
| 45 | X | X | X | | | | 90 | Fixed | Floating-point content reuse |
| 46 | | | X | | | | 91 | Fixed | FXSAVE with extensive SSE and floating-point usage may use incorrect values from the XMM registers |

## 3.7 IA-32 Execution Layer Errata (Sheet 3 of 3)

| No. | IA-32 EL Version | | | | | | Pg. | Status | ERRATA |
|---|---|---|---|---|---|---|---|---|---|
| | 4.3 | 4.4 | 5.3 | 6.5 | | | | | |
| 47 | | | X | | | | 91 | Fixed | Interruption of a loop with SSE may incorrectly restore XMM registers |
| 48 | | | X | | | | 91 | Fixed | Unmasked numeric FP exception in FXTRCT may view wrong FP values |
| 49 | | | X | | | | 92 | Fixed | On rare conditions, FP exceptions shortly after an FCLEX/FNCLEX may view wrong status bits |
| 50 | | | X | | | | 92 | Fixed | An unmasked inexact SSE exception on some instructions may not be restored correctly |
| 51 | X | X | X | | | | 92 | Fixed | SSE exceptions in a hot block may incorrectly set flags |
| 52 | | | X | | | | 93 | Fixed | Multiple exceptions between two code blocks may lead to an incorrect context |
| 53 | | | X | | | | 93 | Fixed | Numeric SSE exceptions could be ignored initially after being unmasked |
| 54 | | | X | | | | 93 | Fixed | Application writing to a guarded page on Windows may fail on access violation |
| 55 | | | X | | | | 93 | Fixed | Job Memory Limit on Windows |
| 56 | | | X | | | | 94 | Fixed | Reloading a modified DLL may fail |
| 57 | | | X | | | | 94 | Fixed | Linux* core file generation |
| 58 | | | X | | | | 94 | Fixed | Linux* EXECVE fails to launch NR file |
| 59 | X | X | X | | | | 94 | Fixed | ptrace returns wrong system-call id |
| 60 | X | X | X | | | | 94 | Fixed | READV/WRITEV overflow |
| 61 | | | X | | | | 94 | Fixed | More precise FP calculation result |
| 62 | | | X | X | | | 95 | Plan Fix | Wrong exception flags in interrupted context |
| 63 | | | X | X | | | 95 | Plan Fix | Wrong CF/AF in interrupted context for LOCK SBB |
| 64 | | | X | X | | | 95 | Plan Fix | Wrong ZF/PF/SF in interrupted context for AAM |
| 65 | | | X | X | | | 95 | Plan Fix | Unaligned RMW instruction interruption handling |
| 66 | | | X | X | | | 95 | Plan Fix | Unexpected access violation on PUSH/POP |
| 67 | | | X | X | | | 95 | Plan Fix | Wrong interrupted EIP on instructions consuming PF |
| 68 | | | | X | | | 96 | Plan Fix | Interruption in unaligned CMPXCHG |
| 69 | | | X | X | | | 96 | Plan Fix | Wrong exception masks in MXCSR |
| 70 | | | X | X | | | 96 | Plan Fix | Wrong flags in interrupted context |
| 71 | | | X | X | | | 96 | Plan Fix | Wrong TOP in interrupted context for SQRTSS |
| 72 | | | X | X | | | 96 | Plan Fix | Applications unexpectedly abort |
| 73 | | | X | X | | | 96 | Plan Fix | Lock instruction with unaligned memory reference |
| 74 | | | | X | | | 97 | Plan Fix | Second 4/8/16-byte unaligned load |
| 75 | | | X | X | | | 97 | Plan Fix | Wrong FP registers value in interrupted context |
| 76 | X | X | X | | | | 97 | Fixed | LOCK NOTW [odd address] negates 4B |
| 77 | | | X | | | | 97 | Fixed | LOCK RMW suspension atomicity break |
| 78 | X | X | X | | | | 97 | Fixed | Flags on CMPXCHGW |
| 79 | X | X | X | | | | 97 | Fixed | Wrong ZF on cmpxchg8b |
| 80 | X | X | X | | | | 98 | Fixed | Flags at interrupt after then/else |
| 81 | X | X | X | | | | 98 | Fixed | DIVPS [m128] interruption crash |
| 82 | X | X | X | | | | 98 | Fixed | Crash on optimization sequence |
| 83 | | | X | | | | 98 | Fixed | Ignored Self Modifying BTX |
| 84 | X | X | X | | | | 98 | Fixed | Lost signal in spin-loop |
| 85 | | | X | | | | 98 | Fixed | Debugger aborts on “fail to attach” |

***Note:***

1. To obtain information on which IA-32 EL version is installed, from the directory that the IA-32 EL binaries are located (/emul/bin, /emul, c:\windows\system32\, and so forth.)
   - In Windows: right-click IA32Exec.bin->Properties->Tab “Version” and look for file Version x.x.xxxx.
   - In Linux* in a command line window, write “libia32x.so -v”.

   The leading number represents the major version number.

## 3.8 Intel® Itanium® 2 Processor (up to 3 MB L3 Cache) Specification Changes

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | SPECIFICATION CHANGES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | |
| | | | | | | | | | | | | | | None for this revision of the Specification Update |

## 3.9 Intel® Itanium® 2 Processor (up to 3 MB L3 Cache) Specification Clarifications

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | SPECIFICATION CLARIFICATIONS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | |
| 1 | X | | | | | | | | | | | | 78 | Error logging of deferred IPIs |
| 2 | X | | | | | | | | | | | | 78 | Branch prediction across the 40-bit boundary |
| 3 | X | | | | | | | | | | | | 78 | PAL_FREQ_RATIOS in the Intel Itanium Architecture Software Developers Manual, revision 2.2 |

## 3.10 Intel® Itanium® 2 Processor (up to 3 MB L3 Cache) Documentation Changes

| No. | Processor Stepping | PAL Version | | | | | | | | | | | Pg. | DOCUMENTATION CHANGES |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | B3 | 7.13 | 7.31 | 7.36 | 7.37 | 7.40 | 7.59 | 7.71 | 7.73 | 7.77 | 7.78 | 7.79 | | |
| | | | | | | | | | | | | | | None for this revision of the Specification Update |

## 3.11 Intel® Itanium® 2 Processor (up to 6 MB L3 Cache) Specification Changes

| No. | Processor Stepping | PAL Version | | | | | | Pg. | SPECIFICATION CHANGES |
|---|---|---|---|---|---|---|---|---|---|
| | B1 | 5.37 | 5.61 | 5.65 | 5.69 | 5.72 | 5.73 | | |
| | | | | | | | | | None for this revision of the Specification Update |

## 3.12 Intel® Itanium® 2 Processor (up to 6 MB L3 Cache) Specification Clarifications

| No. | Processor Stepping | PAL Version | | | | | | Pg. | SPECIFICATION CLARIFICATIONS |
|---|---|---|---|---|---|---|---|---|---|
| | B1 | 5.37 | 5.61 | 5.65 | 5.69 | 5.72 | 5.73 | | |
| 1 | X | | | | | | | 78 | Error logging of deferred IPIs |
| 2 | X | | | | | | | 78 | Branch prediction across the 40-bit boundary |
| 3 | X | | | | | | | 78 | PAL_FREQ_RATIOS in the Intel Itanium Architecture Software Developers Manual, revision 2.2 |

## 3.13 Intel® Itanium® 2 Processor (up to 6 MB L3 Cache) Documentation Changes

| No. | Processor Stepping | PAL Version | | | | | | Pg. | DOCUMENTATION CHANGES |
|---|---|---|---|---|---|---|---|---|---|
| | B1 | 5.37 | 5.61 | 5.65 | 5.69 | 5.72 | 5.73 | | |
| | | | | | | | | | None for this revision of the Specification Update |

## 3.14 Intel® Itanium® 2 Processor (up to 9 MB L3 Cache) Specification Changes

| No. | Processor Stepping | | PAL Version | | | | | | Pg. | SPECIFICATION CHANGES |
|---|---|---|---|---|---|---|---|---|---|---|
| | A1 | A2 | 1.27 | 2.10 | 2.14 | 2.15 | | | | |
| | | | | | | | | | | None for this revision of the Specification Update |

## 3.15 Intel® Itanium® 2 Processor (up to 9 MB L3 Cache) Specification Clarification

| No. | Processor Stepping | | PAL Version | | | | | | Pg. | SPECIFICATION CLARIFICATIONS |
|---|---|---|---|---|---|---|---|---|---|---|
| | A1 | A2 | 1.27 | 2.10 | 2.14 | 2.15 | | | | |
| 1 | X | X | | | | | | | 78 | Error logging of deferred IPIs |
| 2 | X | X | | | | | | | 78 | Branch prediction across the 40-bit boundary |
| 3 | X | X | | | | | | | 78 | PAL_FREQ_RATIOS in the Intel Itanium Architecture Software Developers Manual, revision 2.2 |

## 3.16 Intel® Itanium® 2 Processor (up to 9 MB L3 Cache) Documentation Changes

| No. | Processor Stepping | | PAL Version | | | | | | Pg. | DOCUMENTATION CHANGES |
|---|---|---|---|---|---|---|---|---|---|---|
| | A1 | A2 | 1.27 | 2.10 | 2.14 | 2.15 | | | | |
| | | | | | | | | | | None for this revision of the Specification Update |

## 3.17 Dual-Core Intel® Itanium® 2 Processor 9000 Series Specification Changes

| No. | Processor Stepping | | PAL Version | | Pg. | SPECIFICATION CHANGES |
|---|---|---|---|---|---|---|
| | C1 | C2 | 7.46 | 8.30 | | |
| | | | | | | None for this revision of the Specification Update |

## 3.18 Dual-Core Intel® Itanium® 2 Processor 9000 Series Specification Clarification

| No. | Processor Stepping | | PAL Version | | Pg. | SPECIFICATION CLARIFICATIONS |
|---|---|---|---|---|---|---|
| | C1 | C2 | 7.46 | 8.30 | | |
| 3 | X | X | | | 78 | PAL_FREQ_RATIOS in the Intel Itanium Architecture Software Developers Manual, revision 2.2 |

## 3.19 Dual-Core Intel® Itanium® 2 Processor 9000 Series Documentation Changes

| No. | Processor Stepping | | PAL Version | | Pg. | DOCUMENTATION CHANGES |
|---|---|---|---|---|---|---|
| | C1 | C2 | 7.46 | 8.30 | | |
| 1 | | | X | X | 79 | PAL_MC_ERROR_INJECT err_data_buffer description change |
| 2 | | | X | X | 79 | PAL_MC_ERROR_INJECT procedure err_struct_info - Register File change |

## 3.20 IA-32 Execution Layer Specification Clarifications

| No. | IA-32 EL Version | | | | | | Pg. | SPECIFICATION CLARIFICATIONS |
|---|---|---|---|---|---|---|---|---|
| | 4.3 | 4.4 | 5.3 | 6.5 | | | | |
| 1 | X | X | X | | | | 100 | Aliasing of MMX registers to FP registers |
| 2 | X | X | X | | | | 100 | Floating-point and SSE precision |
| 3 | X | X | X | | | | 100 | CPUID values represent the IA-32 execution layer processor model |
| 4 | X | X | X | | | | 100 | IA-32 execution layer resides in the application virtual address space |
| 5 | X | X | X | | | | 100 | Signal delivery may be postponed during code translation or garbage collection |
| 6 | X | X | X | | | | 100 | Aborting threads could cause other process threads to hang |
| 7 | X | X | X | | | | 101 | Core dump files cannot be produced correctly when an IA-32 process is aborted |
| 8 | X | X | X | | | | 101 | The I/O Privilege Level (IOPL) mechanism is not implemented |
| 9 | X | X | X | | | | 101 | Software interrupts must be supported by the OS |
| 10 | X | X | X | | | | 101 | Intersegment calls require OS mechanism |
| 11 | X | X | X | | | | 101 | Thread creation may be reported incorrectly to the OS |
| 12 | | | X | | | | 101 | Core-dump file may contain Itanium® architecture details |
| 13 | X | X | X | | | | 101 | IA-32 process may hang while generating core-dump file |
| 14 | X | X | X | | | | 101 | DLL unload issue |

***Note:***

1. To obtain information on which IA-32 EL version is installed, from the directory that the IA-32 EL binaries are located (/emul/bin, /emul, c:\windows\system32\, and so forth)
   - In Windows: right-click IA32Exec.bin->Properties->Tab “Version” and look for file Version x.x.xxxx.
   - In Linux in a command line window, write “libia32x.so -v”.

   The leading number represents the major version number.

# 4 Identification Information

## 4.1 Intel® Itanium® 2 Processor Package Marking

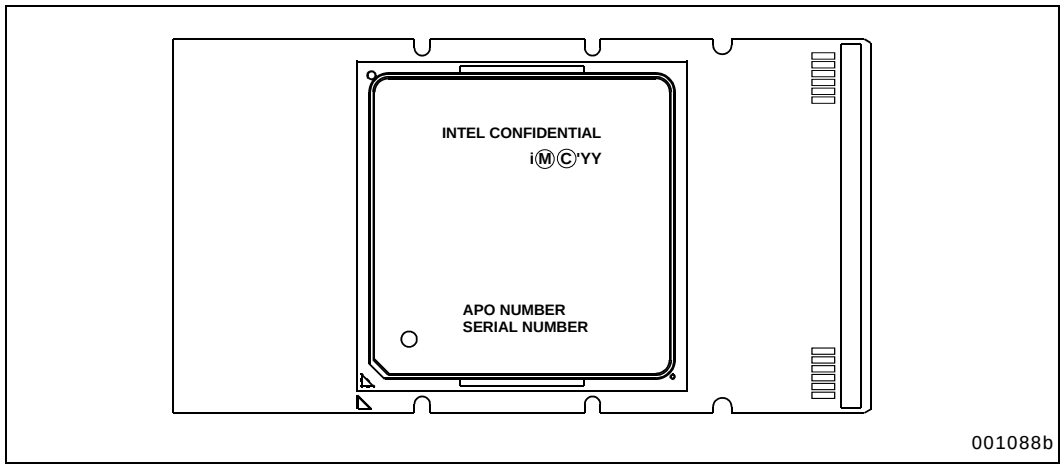
The following section details the processor top-side and bottom-side markings for the Intel® Itanium® 2 processor and is provided as an identification aid. The processor top-side mark for the product is a laser marking on the Integrated Heat Spreader (IHS).

### 4.1.1 Processor Top-Side Marking

Figure 4-1 shows an example of the laser marking on the IHS. The processor top-side mark provides the following information:

- INTEL Brand/ INTEL Product
- Legal Mark
- Assembly Process Order (APO) Number
- Serial Number

**Figure 4-1. Processor Top-Side Marking on IHS**

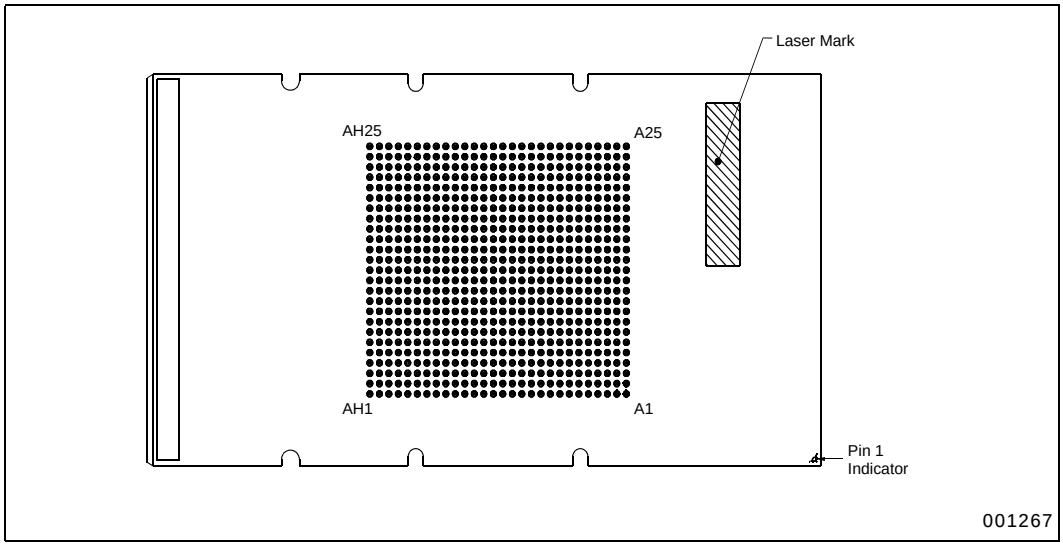



### 4.1.2 Bottom-Side Marking

The processor bottom-side mark for the product is a laser marking on the pin side of the interposer. Figure 4-2 shows the placement of the laser marking on the pin side of interposer. The processor bottom-side mark provides the following information:

- Product ID
- Finish Process Order (FPO) Number
- Serial Number
- S-Spec
- Country of Origin
- 2D Matrix Mark included on Intel® Itanium® 2 processor (up to 6 MB L3 cache) only. Not included on Intel® Itanium® 2 processor (up to 3 MB L3 cache).

**Figure 4-2. Processor Bottom-Side Marking Placement on Interposer**




## 4.2 Dual-Core Intel® Itanium® 2 Processor 9000 Series Package Marking

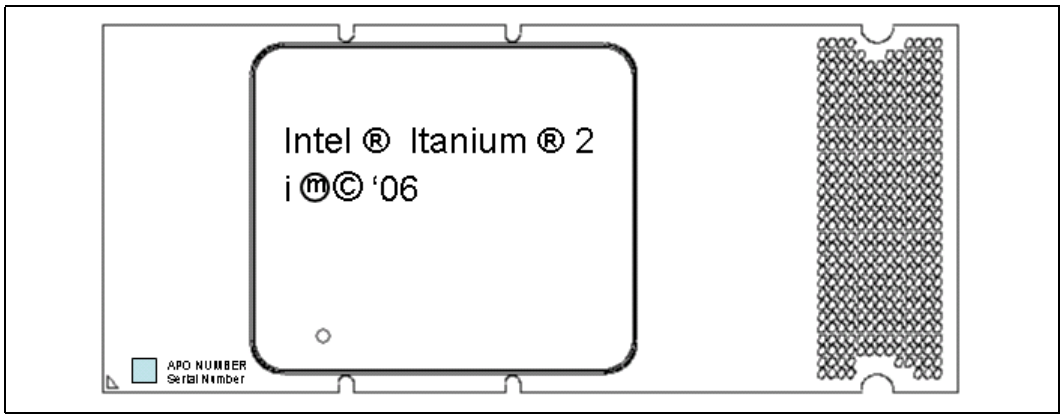
The following section details the processor top-side and bottom-side markings for the Dual-Core Intel® Itanium® 2 Processor 9000 Series processor and is provided as an identification aid. The processor top-side mark for the product is a laser marking on the Integrated Heat Spreader (IHS).

### 4.2.1 Processor Top-Side Marking

Figure 4-3 shows an example of the laser marking on the IHS. The processor top-side mark provides the following information:

- INTEL Brand/ INTEL Product
- Legal Mark
- Assembly Process Order (APO) Number
- Serial Number
- 2-D Matrix

**Figure 4-3. Processor Top-Side Marking on IHS**

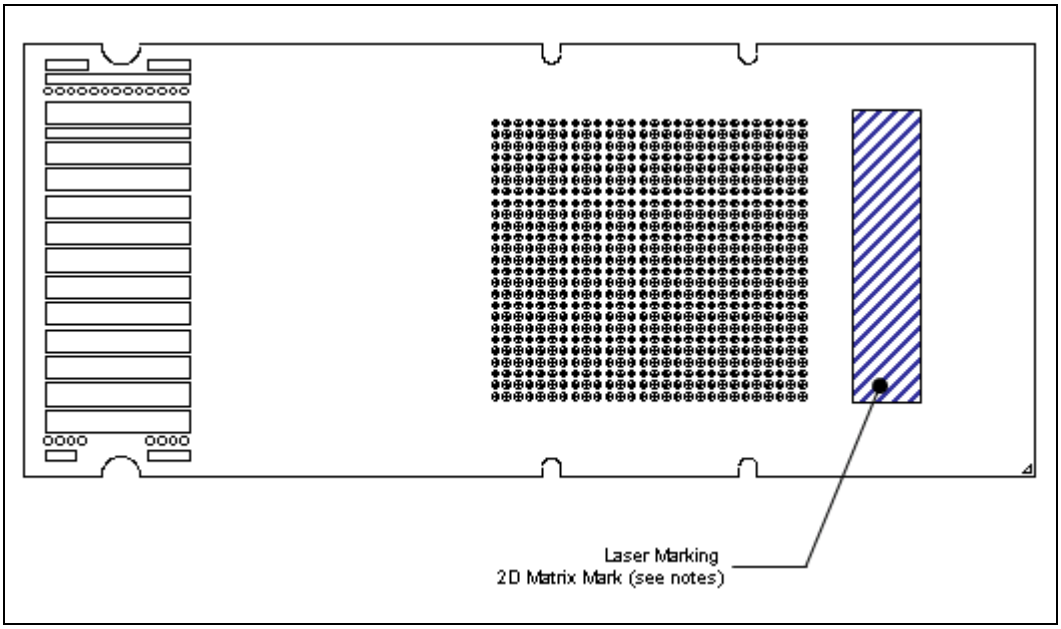


### 4.2.2 Bottom-Side Marking

The processor bottom-side mark for the product is a laser marking on the pin side of the interposer. Figure 4-4 shows the placement of the laser marking on the pin side of interposer. The processor bottom-side mark provides the following information:

- Product ID
- Finish Process Order (FPO) Number
- S-Spec
- 2D Matrix Mark

**Figure 4-4. Processor Bottom-Side Marking Placement on Interposer**



## 4.3 Intel® Itanium® 2 Processor Identification and Package Information

| S-Spec Number | Processor Stepping | CPUID[1] | Speed (MHz) | L3 Size (Mbytes) |
|---|---|---|---|---|
| SL67U | B3 | 001F000704h | 1000/400 | 1.5 |
| SL67V | B3 | 001F000704h | 1000/400 | 3 |
| SL67W | B3 | 001F000704h | 900/400 | 1.5 |
| SL6P5 | B3 | 001F000704h | 1000/400 | 1.5 |
| SL6P7 | B3 | 001F000704h | 1000/400 | 3 |
| SL6P6 | B3 | 001F000704h | 900/400 | 1.5 |
| SL6XF | B1 | 001F010504h | 1500/400 | 6 |
| SL6XE | B1 | 001F010504h | 1400/400 | 4 |
| SL6XD | B1 | 001F010504h | 1300/400 | 3 |
| SL76K | B1 | 001F010504h | 1400/400 | 1.5 |
| SL754 | B1 | 001F010504h | 1000/400 | 1.5 |
| SL7FP | B1 | 001F010504h | 1400/400 | 3 |
| SL7FQ | B1 | 001F010504h | 1600/400 | 3 |
| SL7SD | A1 | 001F020104h | 1300/400 | 3 |
| SL7ED | A1 | 001F020104h | 1500/400 | 4 |
| SL7EC | A1 | 001F020104h | 1600/400 | 3 |
| SL7EB | A1 | 001F020104h | 1600/400 | 6 |
| SL87H | A1 | 001F020104h | 1600/400 | 9 |
| SL7EF | A1 | 001F020104h | 1600/533 | 3 |
| SL8CY | A2 | 001F020204h | 1300/400 | 3 |
| SL8CX | A2 | 001F020204h | 1500/400 | 4 |
| SL8CW | A2 | 001F020204h | 1600/400 | 3 |
| SL8CV | A2 | 001F020204h | 1600/400 | 6 |
| SL8CU | A2 | 001F020204h | 1600/400 | 9 |
| SL8CZ | A2 | 001F020204h | 1600/533 | 3 |
| SL8JK | A2 | 001F020204h | 1660/667 | 6 |
| SL8JJ | A2 | 001F020204h | 1660/667 | 9 |
| L98T[2] | C1 | 0020000504h | 1600/533 | 24 |
| L9DF | C1 | 0020000504h | 1600/533 | 24 |
| L9DG | C1 | 0020000504h | 1600/533 | 18 |
| L9DE | C1 | 0020000504h | 1420/533 | 12 |
| L9BW | C1 | 0020000504h | 1400/400 | 12 |
| L9DH[4] | C1 | 0020000504h | 1600/533 | 8 |
| L9DJ[3] | C1 | 0020000504h | 1600/533 | 6 |
| L9P7[2] | C2 | 0020000704h | 1600/533 | 24 |
| L9PG | C2 | 0020000704h | 1600/533 | 24 |
| L9P8 | C2 | 0020000704h | 1600/533 | 18 |
| L9PB | C2 | 0020000704h | 1420/533 | 12 |
| L9PC | C2 | 0020000704h | 1400/400 | 12 |
| L9P9[4] | C2 | 0020000704h | 1600/533 | 8 |
| L9PA[3] | C2 | 0020000704h | 1600/533 | 6 |

***Notes:***

1. The CPUID column in this table indicates the contents of bits 39:0 of CPUID Register 3. Bits 63:40 of this register are reserved. The Family ID for the Intel® Itanium® 2 processor is 0x1F. The Family ID for the Dual-Core Intel® Itanium® 2 Processor 9000 Series is 0x20.
2. Supports Lockstep operation.
3. Single Core only and does not support hyper-threading technology.
4. Does not support hyper-threading technology.

| Abbreviation | PAL Version[1] | Processor Stepping |
|---|---|---|
| Itanium® 2 Processor (up to 3 MB L3 cache) | 7.13 | B3 |
| | 7.31 | B3 |
| | 7.36 | B3 |
| | 7.37 | B3 |
| | 7.40 | B3 |
| | 7.59 | B3 |
| | 7.71 | B3 |
| | 7.73 | B3 |
| | 7.77 | B3 |
| | 7.78 | B3 |
| | 7.79 | B3 |
| Itanium 2 Processor (up to 6 MB L3 cache) | 5.37 | B1 |
| | 5.61 | B1 |
| | 5.65 | B1 |
| | 5.69 | B1 |
| | 5.72 | B1 |
| | 5.73 | B1 |
| Itanium 2 Processor (up to 9 MB L3 cache) | 1.27 | A1 |
| | 2.10 | A1, A2 |
| | 2.14 | A1, A2 |
| | 2.15 | A1, A2 |
| Dual-Core Intel® Itanium® 2 Processor 9000 Series | 7.46 | C1, C2 |
| | 8.30 | C1, C2 |

***Notes:***

1. Please refer to the applicable PAL release notes for information regarding changes in each PAL release.



# 5 Limited Support for Mixed Steppings

Intel Corporation limits support for mixed steppings of the Intel® Itanium® 2 processor (up to 9 MB L3 cache). The following list describes the requirements to support mixed steppings:

- Mixed steppings of processors are only supported with the following paired combinations of A1 and A2 steppings of the Intel® Itanium® 2 processor with up to 9 MB L3 cache, identified by the package S-Spec numbers (see the Intel® Itanium® 2 Processor Identification and Package Information table for details):
  - SL7SD and SL8CY
  - SL7ED and SL8CX
  - SL7EC and SL8CW
  - SL7EB and SL8CV
  - SL87H and SL8CU
  - SL7EF and SL8CZ
- While Intel has done nothing to specifically prevent processors operating at differing frequencies from functioning within a multiprocessor system, there may be uncharacterized errata that exist in such configurations. Intel does not support such configurations. In mixed stepping systems, all processors must operate at identical frequencies (that is, the highest frequency rating commonly supported by all processors).
- While there are no known issues associated with the mixing of processors with differing cache sizes in a multiprocessor system, and Intel has done nothing to specifically prevent such system configurations from operating, Intel does not support such configurations since there may be uncharacterized errata that exist. In mixed stepping systems, all processors must be of the same cache size.
- While Intel believes that certain customers may wish to perform validation of system configurations with mixed frequency or cache sizes, or voltages and that those efforts are an acceptable option to our customers, customers would be fully responsible for the validation of such configurations.
- Intel requires that the latest version of PAL code be used in the system firmware. Any system firmware that is not using the latest version of PAL is considered by Intel to be operating out of specification.
- The workarounds identified in this and following specification updates must be properly applied to each processor in the system. Certain errata are specific to the multiprocessor environment. Errata for all processor steppings will affect system performance if not properly worked around. Also see the processor Identification and Package Information section for additional details on which processors are affected by specific errata.

While there are no known issues associated with the mixing of processors with differing voltages in a multiprocessor system, and Intel has done nothing to specifically prevent such system configurations from operating, Intel does not support such configurations since there may be uncharacterized errata that exist. In mixed stepping systems, all processors must be of the same voltage.

# 6 Errata (Processor and PAL)

## 1. IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED does not count predicated off instructions

**Problem:** The event monitor count for instructions retired (IA64_INST_RETIRED and IA64_TAGGED_INST_RETIRED) does not include the predicated off instructions.

**Implication:** The IA64_INST_RETIRED/IA64_TAGGED_INST_RETIRED performance monitoring events may report an incorrect count.

**Workaround:** Add the PREDICATE_SQUASHED_RETIRED event monitor count to the IA64_INST_RETIRED and/or the IA64_TAGGED_INST_RETIRED event monitor count to get the intended results.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

## 2. Performance Monitor Interrupt raised when freeze bit is written to Performance Monitoring Counter register

**Problem:** The Performance Monitor Freeze (PMC[0].fr) bit within the Performance Monitoring Counter (PMC) register is used to stop performance event monitoring. This can be set by software or by an event counter overflow. Due to this erratum, the processor may raise a Performance Monitor Interrupt (PMI) when the freeze bit is set by software, even when the Performance Monitor Overflow Interrupt (PMC.oi) bit is not enabled and no overflow has occurred.

**Implication:** The processor may generate a PMI when it's not expected to do so.

**Workaround:** The interrupt service routine (ISR) needs to account for the spurious interrupt even if no performance monitor overflow is indicated.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

## 3. Priority agent requests with unit mask of I/O not counted

**Problem:** The system bus allows for the BPRI# signal to be asserted one cycle before an ADS# is driven by the priority agent, provided no BREQ# pins are driven by the processor. Priority agent requests exhibiting this behavior are not counted by the system bus performance monitoring events when using a unit mask of 'I/O'.

**Implication:** The system bus performance monitoring events may report an incorrect count in this case.

**Workaround:** Measure the bus transactions for all bus masters (unit mask= 'ANY') and subtract from it the sum of the corresponding bus transactions on each local processor (unit mask= 'SELF').

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 4. Incorrect fault reporting on move to/from the RNAT or BSPSTORE application registers

**Problem:** Incorrect faulting behavior may be experienced under the following conditions:

1. A `mov.imm` (move immediate) to the ar.rsc register is executed in the same instruction bundle (or the next bundle with no intervening stop bits) as a mispredicted branch.
2. The mispredicted branch path includes another `mov.imm` to the same ar.rsc register, and is within two issue groups or less of the (mispredicted) branch instruction. This instruction is not executed. Also, the value moved to the rsc.mode field must be different than the value moved to rsc.mode in the `mov.imm` in step 1.
3. The correct branch path is then taken and includes a move to/from the ar.rnat or ar.bspstore registers, within the first bundle (or second bundle with no intervening stop bit) of the correct branch instruction.



**Implication:** When the above conditions line up (and there are no stalls or cache misses), the instruction in step 3 (move to/from ar.rnat or ar.bsp) uses the rse.mode value from the mov.imm in the mispredicted branch path instead of from instruction in step 1. As a result, there may be incorrect faulting behavior – an illegal opcode fault is missed (if rse.mode!= 0) or falsely indicated (if rse.mode = 0) and may result in inconsistent system behavior. This erratum has only been observed in a system validation environment.

**Workaround:** Use one of the following workarounds:

1. Use the register form of the move instruction or;
2. Ensure there is a stop bit between any `mov.imm` instruction to/from the ar.rsc registers and any subsequent branch instruction or;
3. Ensure that there is a stop bit between a “label” (branch target) and a subsequent move to/from ar.rnat/ar.bspstore.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 5. Power good deassertion affects boundary scan testing

**Problem:** Deassertion of the PWRGOOD signal during boundary scan testing prevents the correct operation of the sampling functionality in the EXTEST and SAMPLE/PRELOAD JTAG commands.

**Implication:** As a result of this erratum the boundary scan chain function is disabled and will stop shifting data when the PWRGOOD signal is low.

**Workaround:** Keep the PWRGOOD signal asserted during boundary scan testing.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 6. IA-32: CPUID instruction returns incorrect L3 cache size

**Problem:** The IA-32 `CPUID` instruction will always report the L3 cache size as 3 MB regardless of the actual size of the L3 cache.

**Implication:** IA-32 applications using the IA-32: `CPUID` instruction cannot rely on the cache size reported by this instruction. Native Intel® Itanium® architecture-based applications are not affected by this erratum and can access this information via the processor `CPUID` registers.

**Workaround:** Within the Linux *operating system (OS) environment, the ‘/proc/cpuinfo’ file contains this information. Within the Microsoft* OS environment this information is available through Windows API calls.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 7. Performance Monitoring Event counters may be incorrect when using Instruction Address Range checking in fine mode

**Problem:** For performance monitoring events that use Instruction Address Range Matching set to ‘Fine Mode’ (PMC: 14, bit 13 = 1), the address matching capability will be inconsistent and may yield incorrect results.

**Implication:** Due to this erratum the results of an event counter while using ‘Fine Mode’ may not be correct.

**Workaround:** Use normal mode when using Instruction Address Range checking.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 8. Possible deadlock condition after ptc.g is issued on two-way system

**Problem:** In a two processor system, a `ptc.g` instruction is issued on processor A. The execution of the `ptc.g` on processor A blocks the completion of a semaphore upon which processor B is waiting to become available. Concurrently processor B is issuing a long series of loads and stores with one or more instructions being retried or involves system memory access before being retired. Processor B’s L2 cache entry queue, denoted as OzQ, is full and does not allow the `ptc.g` operation from processor A, entry into the L2 OzQ for completion. The `ptc.g` request will be presented again in three clock cycles. If processor B continues to execute a code sequence such that the L2 cache OzQ entries continue to be taken by other load/stores, then the `ptc.g` operation must continue to wait.

**Implication:** Due to this erratum, the system may deadlock while waiting for the `ptc.g` to be completed. Any break in, or completion of the code loop on processor B, including system interrupts, that allows the `ptc.g` operation to enter the L2 cache OzQ on processor B will be enough to release the deadlock condition. Additional processors will also change the time cycle necessary for this event to occur. This issue has only been observed during Random Instruction Testing in a system validation environment.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 9. EPC, mov ar.pfs and br.ret instructions may combine to yield incorrect privilege level

**Problem:** Due to certain internal timing and microarchitectural conditions, OS calls that return to user space from privilege code promote pages using a `br.ret` instruction, may not have the expected privilege level.

Using the following code sequence as an example:

```
<change of privilege level>          //epc on promote page; or br.ret
mov ar.pfs, [value];                 //new pfs value has ppl < cpl
br.ret;;
```

In this case, the `br.ret` is specified to not change the privilege level (`pl`) since the `br.ret` is asking to promote privilege to a numerically lower level. Current processor steppings may change current privilege level (`cpl`) to the `pl` at the beginning of the <change of privilege level>.

**Implication:** This erratum would result in having the `cpl` demoted and the user space application may not receive the correct privilege level. Privilege code promote page usage is limited and controlled by the OS. This issue has only been observed during random instruction testing in a system validation environment.

**Workaround:** Use one of the following workarounds:

1. Use an return from interrupt (`rfi`) instruction instead of `br.ret` to return from privilege code promote pages.
2. Insert a useless call-to-next bundle in all paths leading to a demoting `br.ret`.
3. PAL version 7.01 and above, have a workaround for this issue and it is enabled by default. The OS may implement one of the previous workarounds or a check mechanism, such that this PAL workaround can be disabled. Please review the PAL Release notes for details on the implementation of this workaround.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 10. Removal of WAW hazard may lead to undefined result

**Problem:** Due to internal conditions an allowed WAW dependency may become a WAW hazard under the following circumstances:

- A move to the AR.PFS register is followed by a BR.CALL and both are executed in the same issue group, or
- A move to the AR.EC register is followed by a BR.RET and both are executed in the same issue group.

These combinations of instructions are legal WAW memory dependencies if one of the operations is predicated off. If preceding instructions (as indicated above) combine to change the predication on the BR.CALL or BR.RET from predicated true to predicated false, the processor may mistakenly decide the WAW hazard is still present and fail to recognize that the WAW has been removed which may result in an undefined value for ar.pfs or ar.ec.

The following code sequence demonstrates this issue:

```
p15 = 1;
;;
mov ar.pfs = R[x];
ld.c R[y] = [m]; //causes R[y] to be reloaded.

cmp.eq p15, p16 = R[y], R0;
(p15) br.call;
```

The RAW dependencies on `ld.c` to `cmp` and `cmp` to branch are legal. When the processor executes the issue group, the WAW hazard is present and the PFS results are undefined. If the `ld.c` misses the advanced load address table (ALAT), the `cmp` to branch will be re-executed, the new result of the `ld.c` causes the p15 value to change to false and thus eliminate the WAW. Then the processor may fail to recognize that the WAW has been removed.

**Implication:** An application may hang or signal an exception fault under these circumstances. The affected code sequence is not known by Intel to be generated in any current compiled code or exist in any current OS.

**Workaround:** Separate the predicate producing instruction from its consumer with a stop (as recommended in the *Intel® Itanium® Architecture Software Developer's Manual*, Volume 1: Application Architecture) or change the predication sequence to assure mutually exclusive predication of the instructions in the WAW dependency.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 11. Unexpected data debug, data access or dirty bit fault taken after rfi instruction

**Problem:** A fault may be taken after a `rfi` instruction has been executed under the following circumstances. The IPSR.da or IPSR.dd bits are set to disable data debug/data access/dirty bit faults for the first Intel® Itanium® processor system environment restore instruction. This is followed by an `rfi` instruction. The `rfi` instruction is followed by additional instructions that generate register stack engine (RSE) activity (`alloc`, `flushrs`, `br.ret`). The processor will see the RSE activity as valid Itanium system instructions and clear the ipsr.da/dd bits and this may result in an unexpected data debug, data access or dirty bit fault at the target of the `rfi`.

**Implication:** Due to this erratum an unexpected fault may be generated after an `rfi` instruction has been executed. This may slow the transition of the system into the Itanium system environment and log un-necessary errors.

**Workaround:** Separate the `rfi` from the RSE generating instruction by four issue groups of `nop` instructions.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 12. Incorrect privilege level may be granted if a failed speculation check precedes a privilege level change

**Problem:** A failed speculation check instruction (`chk.s/chk.a/fchkf`) that is followed by a privilege change operation may result with the incorrect privilege level for instructions in the issue group of the privilege level change and beyond. The privilege changing instruction must occur within two clock cycles of the failed speculation check.

**Implication:** As a result of this erratum, the speculation check recovery code and subsequent instructions may have an incorrect privilege level.

**Workaround:** Do not use speculation near privilege changing instructions. The workaround for this erratum is to escalate failed speculation checks (speculation check re-steers) to the OS for recovery. This workaround is included in PAL version 7.01 and above.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 13. Floating-point instructions take a floating-point trap before Unimplemented Instruction Address trap

**Problem:** A floating-point instruction that causes a floating-point trap and is the last instruction at the top of the physical address space should flag an Unimplemented Address trap before the floating-point trap.

**Implication:** The correct trap is flagged but only after the floating-point trap is taken first.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 14. PAL_MC_ERROR_INFO does not return an address for certain double bit ECC memory errors

**Problem:** PAL_MC_ERROR_INFO will report the address for the source of a double bit ECC memory error. However, under the conditions that the data with a 2x ECC error was prefetched to the L2 cache and later filled into the L1 cache, the source address will not be available.

**Implication:** PAL_MC_ERROR_INFO will not be able to report the address of a double bit ECC error in this case. Double bit errors that are consumed in this scenario will be not be recoverable.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 15. PAL_CACHE_READ and PAL_CACHE_WRITE return incorrect status for L1I cache access

**Problem:** The PAL_CACHE_READ and PAL_CACHE_WRITE procedures should return a status value of '–7' (which indicates this operation is not supported for this *cache_type* and *level*) when attempting to read or write to/from the L1I (instruction) and L1D (data) cache. When these procedures attempt to access the L1I cache an incorrect status value will be returned.

**Implication:** Due to this erratum, using these PAL procedures to access the L1I cache will result in the return of an incorrect status value, implying that the L1I cache is readable/ writeable by these PAL procedure calls.

**Workaround:** Do not use these PAL procedures to access the L1D and L1I caches.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 16. Unpredictable behavior if the system is awakened from low power mode by an MCA

**Problem:** If the system is in low power mode and an machine check abort (MCA), BERR# or BINIT# is signaled, the PALE_CHECK handler will be called to process the error condition. However, PALE_CHECK does not disable low power mode so that it can continue execution. As soon as PALE_CHECK attempts to drain the processor queues, the system may re-enter low power mode. This may cause incomplete handling of the error event and potentially, intermittent continuation of the same event during later signaled BINIT# events.

**Implication:** The processor can appear to be trapped in low power mode and/or system behavior may be unpredictable.

**Workaround:** Do not use low power mode or call the following PAL procedures: PAL_HALT, PAL_HALT_LIGHT or PAL_HALT_LIGHT_SPECIAL.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 17. The system may lose an interrupt when SAL_CHECK reads the IVR

**Problem:** The PAL_REGISTER_INFO procedure returns an incorrect value to indicate that reading the Interrupt Vector Register (IVR), CR65 (Configuration Register 65) has no side effects. Based on this incorrect return value, when SAL_CHECK reads the IVR while saving system state data to NVRAM, a pending interrupt may be allowed to proceed before the current process has been completed.

**Implication:** The SAL_CHECK procedure relies on the return values of PAL_REGISTER_INFO to know which ARs and CRs are safe to read and save off. Due to this erratum, the SAL_CHECK reads the IVR, and consequently causes the corresponding bit in the IRR to be cleared and the ISR to change. The results of the interrupt routine currently being executed may be lost.



**Workaround:** After calling PAL_REGISTER_INFO with *info_request* = 3, System Abstraction Layer (SAL) can force the correct return value for CR65 by setting bit 1 of *reg_info_2* to a value of one.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 18. A bus MCA nested within a recoverable or firmware-corrected bus MCA may not be handled correctly

**Problem:** During the processing of a non-fatal bus MCA, if a second bus MCA is received the second MCA may be missed.

**Implication:** A bus MCA received in this scenario may be missed and result in unpredictable system behavior. If the first MCA is fatal, system behavior remains correct.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 19. PAL reset sequence performed after a recovery check may result in incorrect system behavior

**Problem:** The PAL early self-test sequence performed after a recovery check may not properly serialize outstanding memory transactions.

**Implication:** As a result of this erratum, memory transactions that are outstanding at the point of transition from the recovery check handler to PAL may cause a deadlock condition and possibly hang the processor.

**Workaround:** SAL can call the PAL_MC_DRAIN procedure before returning to PAL from recovery check to ensure that outstanding transactions have completed.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 20. PAL_HALT_LIGHT_SPECIAL provides PAL_HALT functionality

**Problem:** The PAL_HALT_LIGHT_SPECIAL procedure does not issue the stop grant acknowledge special bus cycle.

**Implication:** PAL_HALT_LIGHT_SPECIAL behavior will be the same as PAL_HALT.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 21. PAL_TEST_PROC may access memory with the UC attribute

**Problem:** The ‘mem_attr’ self-test in PAL_TEST_PROC may access memory with the UC attribute, even though the ‘attributes’ parameter does not allow UC access.

**Implication:** PAL_TEST_PROC may access uncacheable memory that may not be supported in some systems.

**Workaround:** Set bit 44 of the PAL_TEST_PROC procedure self-test control word (*st_control*) to ‘1’ to skip the ‘mem_attr’ self-test.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 22. L2 single bit data error promoted to MCA continues to flag a CMCI

**Problem:** With correctable machine check interrupt (CMCI) to MCA promotion enabled and an L2 single bit ECC data error occurs, an MCA is signaled but the CMCI continues to be raised. After the MCA is completed and the system calls the PAL_MC_RESUME procedure, a CMCI is raised if PSR.i = 1 (respond to external interrupts) and the CMCV.m = 0 (CMCI interrupts are pended).

**Implication:** A CMCI continues to be signaled on L2 1x ECC data errors, even if CMCI to MCA promotion is enabled.

**Workaround:** When enabling CMCI to MCA promotion, mask CMCIs by saving the state of CMCV.m then set CMCV.m = ‘1’. Restore the original state of CMCV.m when disabling promotion.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 23. PAL_TEST_PROC requires specific tests be performed for correct operation

**Problem:** PAL_TEST_PROC self-test requires three specific tests be performed, otherwise the PAL procedure may report false failures or unexpected behavior.

**Implication:** The PAL_TEST_PROC procedure must perform the virtual hash page table (VHPT) test (bit 34), late floating-point test (bit 41) and RSE test (bit 45). Otherwise the system may have unexpected behavior or false test failures may be indicated.

**Workaround:** Bits 34, 41 and 45 in the PAL_TEST_PROC self-test control word (*st_control*) should be left at the default settings of '0' so these tests are performed.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 24. PAL_TEST_INFO may return incorrect data for invalid test parameters

**Problem:** The PAL_TEST_INFO procedure may return incorrect data or status if the input arguments are not valid or are out of range for a given parameter.

**Implication:** Calling the PAL_TEST_PROC procedure with invalid inputs may result in incorrect data and/or status instead of indicating invalid arguments.

**Workaround:** Ensure that PAL_TEST_INFO input parameters are valid and within the argument's range.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 25. PAL_CACHE_INIT may not function properly if levels of the cache hierarchy are specified

**Problem:** PAL_CACHE_INIT does not function properly when caches are selected individually.

**Implication:** A call to initialize the L1D cache may hang the processor and a call to initialize any other cache structure may fail and return an error.

**Workaround:** Call the PAL_CACHE_INIT procedure with level = –1 to initialize all caches.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 26. PAL_SET_TIMEOUT may have an unexpected result when time-out = 0

**Problem:** Setting the input parameter time-out = 0 will disable the processor watchdog timer feature.

**Implication:** Calling PAL_SET_TIMEOUT with time-out = 0 disables the internal processor time-out function.

**Workaround:** Do not set the time-out parameter to '0'.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 27. Concurrent MCAs that signal a BERR may not set PSP.bc correctly

**Problem:** In the case of concurrent MCAs that should result in BERR assertion, the PALE_CHECK handler may not set the PSP.bc (bus check error) bit before handing off to SAL.

**Implication:** As a result of this erratum, PAL_MC_ERROR_INFO will indicate that a bus error occurred, but the PSP at hand-off to SAL_CHECK will not.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 28. PAL_PLATFORM_ADDR may return an error if bit 63 is set

**Problem:** PAL_PLATFORM_ADDR should ignore bit 63 of the *address* argument. If this PAL procedure is called with bit 63 set to '1' in the *address* argument, the procedure incorrectly returns status = –2 (invalid argument).

**Implication:** Due to this erratum, calling PAL_PLATFORM_ADDR with bit 63 of the address set to '1' will return a status of 'invalid argument'.

**Workaround:** Bit 63 should be set to '0' when calling the PAL_PLATFORM_ADDR procedure to avoid this issue.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



### 29. PAL_TEST_PROC may overwrite predicate registers

**Problem:** PAL_TEST_PROC may overwrite predicate registers pr4 and pr5, which should be preserved by the procedure.

**Implication:** PAL_TEST_PROC may modify pr4 or pr5, resulting in undefined behavior.

**Workaround:** Code calling this PAL procedure can save and restore these predicate registers around the PAL_TEST_PROC procedure.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 30. Recovery check fails if PAL_B is not found

**Problem:** SAL may not be able to complete a recovery check when no PAL_B is present. The I/O port address, interrupt block and other features may not be available for SAL when recovery check is entered from PAL_A_SPEC.

**Implication:** Recovery check may fail if PAL_B is not available or is invalid.

**Workaround:** Ensure that the firmware interface table (FIT) entry for PAL_B points to a valid and correct version of PAL_B.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 31. PAL procedure calls may have unexpected results if an incorrect PAL_B version is used

**Problem:** PAL procedures that call PAL_B may not provide the expected results if the first PAL_B entry in the FIT points to an incorrect version of PAL_B.

**Implication:** PAL procedures may fail if the PAL_B entry in the FIT is for an incorrect version.

**Workaround:** Ensure that the FIT entry for PAL_B points to the correct version.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 32. Late self-test may have unexpected results during concurrent processor tests

**Problem:** While running PAL_TEST_PROC concurrently on more than one processor and the processors happen to access the same memory address space, a snoop may cause the ALAT test to fail.

**Implication:** If a processor self-test procedure is using the same memory space for concurrent processor testing, the ALAT test may fail and cause one processor to enter a spin loop.

**Workaround:** The ALAT test can be bypassed by setting bit 46 of the PAL_TEST_PROC self-test control word to ‘1’.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 33. PAL_TEST_PROC may cause unexpected system behavior

**Problem:** The PAL_TEST_PROC ‘late floating-point load/store test’ may overwrite the fr2-fr5 and fr30-fr31 floating-point registers and the Bank 0 gr16–gr23 general registers may be overwritten by the ALAT, VHPT, translation lookaside buffer (TLB) and memory attributes tests.

**Implication:** PAL_TEST_PROC may corrupt the following registers: Bank 0 gr16–gr23 (general registers) and the fr2– fr5, fr30–fr31 (floating-point registers).

**Workaround:** Use different registers or save/restore the contents before/after running PAL_TEST_PROC.

Using the self-test control word of the PAL_TEST_PROC procedure, set the following bits to ‘1’: To avoid corrupting the Bank 0 general registers do not run the ALAT (bit 46), VHPT (bit 35), TLB (bit 33) and mem_attr (bit 44) tests. To avoid corrupting the floating-point registers do not run the late_fp_ld_st (bit 40) test.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 34. PAL halt procedures may overwrite predicate registers

**Problem:** Predicate registers pr1, pr2 and pr3 may be overwritten by the PAL_HALT, PAL_HALT_LIGHT and PAL_HALT_LIGHT_SPECIAL procedures.

**Implication:** As a result of this erratum, pr1, pr2 and p3 may be overwritten.

**Workaround:** Save and restore the predicate registers, as needed when calling these PAL procedures.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 35. Two resets may be necessary to leave TAP test mode

**Problem:** After accessing the test access port (TAP), issuing a RESET# may result in the processor entering an idle state instead of beginning normal operation. Signaling a second RESET# may be necessary to properly reinitialize the system under these conditions.

**Implication:** Due to this erratum, a second RESET# may be required to properly reinitialize the processor after the TAP port has been accessed. Normal system operation and boot process is not affected.

**Workaround:** Issue two resets to properly reinitialize the processor after accessing the TAP port.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 36. IA-32 instruction pointers may be overwritten under certain boundary conditions

**Problem:** Under certain internal conditions involving branch prediction and multiple branch instructions, IA-32 instruction pointers may be overwritten and result in IA-32 instructions being executed out of order or incorrectly. An affected code sequence would have consecutive branch instructions that have started execution before being cancelled.

**Implication:** Due to this erratum, IA-32 instruction pointers may be overwritten resulting in incorrect IA-32 instruction execution.

**Workaround:** A workaround for this erratum is included in PAL version 7.31.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 37. Initialization and ETM recovery may overwrite branch register

**Problem:** PAL INIT recovery code may overwrite br0, when it saves the system environment to the min-state save area. This erratum also affects the recovery path of an enhanced thermal management (ETM) alert that is generated while a system is in a low power mode.

**Implication:** INIT and ETM recovery code may overwrite br0, which prevents recovery with PAL_MC_RESUME and may result in unexpected system behavior.

**Workaround:** PAL version 7.31 fixes this issue.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 38. PAL procedures may not save predicate register 3

**Problem:** The following PAL procedures may not properly save and restore predicate register pr3. The affected PAL procedures are:

PAL_CACHE_INIT, PAL_CACHE_LINE_INIT, PAL_CACHE_READ, PAL_CACHE_WRITE, PAL_CAR_INIT, PAL_COPY_INFO, PAL_COPY_PAL, PAL_PROC_SET_FEATURES, PAL_TEST_PROC

**Implication:** Predicate register 3 may be overwritten by the PAL procedures listed above.

**Workaround:** Save and restore pr3, as needed, when calling the aforementioned PAL procedures.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



### 39. PAL_CACHE_INFO procedure may return undefined value

**Problem:** The PAL_CACHE_INFO procedure could return an invalid value in the config_info_1 '*at*' (cache memory attributes) field. When requesting information for the L2 and L3 cache, the '*at*' field may contain the value of 2, which is undefined.

**Implication:** The PAL_CACHE_INFO procedure, *cache memory attributes* field may return an undefined value.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 40. PAL_HALT_LIGHT procedure may generate a spurious Performance Monitor Interrupt

**Problem:** The PAL_HALT_LIGHT procedure may not properly set the value of the PMV.m bit on return from a low power state and as a result, a spurious PMI may be generated.

**Implication:** A spurious PMI may be indicated when using the PAL_HALT_LIGHT procedure.

**Workaround:** Set the PMV.m bit to '1' (to mask PMIs) before calling PAL_HALT_LIGHT. Set the PMV.m bit to '0' on return from the PAL_HALT_LIGHT procedure.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 41. Unexpected system behavior after PAL_CACHE_FLUSH is executed

**Problem:** The PSR.ic bit is not restored after the PAL_CACHE_FLUSH procedure is executed with *cache_type* = 2. This may result in unexpected behavior when an interrupt is received after calling PAL_CACHE_FLUSH.

**Implication:** The system may not respond to interrupts as expected after PAL_CACHE_FLUSH is executed with *cache_type* = 2.

**Workaround:** Save and restore the PSR.ic bit as necessary, before and after calling the PAL_CACHE_FLUSH procedure.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 42. PAL_TEST_PROC may not properly report self-test status

**Problem:** In the case that some PAL_TEST_PROC self-test functions fail, the *test_status* field may not indicate which self-test function has failed. Instead the failed test function may be raised as an initialization failure and the procedure will enter an infinite loop.

**Implication:** The PAL_TEST_PROC procedure may enter an infinite loop as a result of some failed self-tests, instead of operating in a functionally restricted manner.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 43. PSR.ri may not reflect the correct slot upon entrance to the unimplemented address fault handler

**Problem:** In the case of an `rfi` instruction that targets an instruction in slot 1 or 2 and the interrupt instruction pointer (IIP) points to an unimplemented physical address, the PSR.ri may point to slot 0 instead of slot 1 or 2 as expected. The required conditions to expose this erratum are: The processor is in physical address mode (PSR.it=0) and the IIP points to a physical memory address that is unimplemented.

**Implication:** When the processor attempts to execute on the indicated instruction bundle an unimplemented address fault will be taken and the restart instruction will indicate slot 0. Since no instruction in slot 0, 1, or 2 is executable under these conditions, there is no useful information lost when the unimplemented address fault is taken.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 44. WC and WB memory attribute aliasing combine with FC and may cause processor live-lock

**Problem:** Under certain conditions involving write coalescing (WC) stores and the execution of a flush cache (`fc`) instruction, the `fc` may not be able to proceed until the WC buffers have been emptied, resulting in a live-lock condition.

The live-lock is armed when one or more WC stores (st [A]) occur and allocate space in the processor's WC buffer. A store or load (st/ld [B]) with a writeback (WB) memory attribute is issued followed immediately by an `fc` (`fc[C]`) instruction. The `fc` is targeted to a virtual address with the same physical address as address [A], but with a WB memory attribute instead of WC. If address [B] shares the same physical address bits 14:7 with the flush cache target address [C], then the processor may live-lock.

**Implication:** This memory attribute aliasing (MAA) scenario is likely to occur for a short time in OS code page tear down or where a code page was previously accessed with the WC attribute, but is now implicitly considered to have WB attributes because memory translation has been disabled (PSR.dt=0).

Documented in the *Intel® Itanium® Architecture Software Developer's Manual*, Volume 2, Section 4.4.11, as part of the process to properly transition to a new memory attribute, an `fc` instruction should be issued to flush the WC buffers. However, the text also states that a memory fence (`mf`) instruction should precede the `fc` instruction. Properly following this transition procedure will be sufficient to avoid the live-lock condition.

**Workaround:** Precede `fc` instructions with `mf` instructions where WC buffers may be non-empty.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 45. Improper use of memory attribute aliasing may lead to out of order instruction execution

**Problem:** An `fc` instruction is issued to a virtual memory address that has been aliased as uncacheable (UC). This is immediately followed by a load/store to a WB memory address that points to same physical memory address that is targeted by the `fc`. Due to internal conditions, the load/store may be filled from the L2 cache rather then being filled from memory after the `fc` has been completed.

**Implication:** Using MAA in this manner requires the proper transitioning sequence as noted in the *Intel® Itanium® Architecture Software Developer's Manual*, Volume 2, Section 4.4.11. Under these conditions, the order of operations observed directly on the system bus (by using a logic analyzer for example) may appear to be out of order, however there is no functional impact because the result of instruction execution will always be correct internally.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 46. FPSWA may not set the Denormal status flag correctly

**Problem:** In some cases when the Floating-Point Software Assistant (FPSWA) handles the following floating-point operation using the specified floating-point class/subclass types, the FPSWA may not return the correct Denormal/Unnormal (D) status flag setting in the Floating-Point Status Register (FPSR.sf0:8).

The affected operation is: Infinity * unnormalized number - Infinity = QNaN Indefinite.

**Implication:** As a result of this erratum, the FPSWA may indicate a Denormal/Unnormal exception fault where none has occurred.

**Workaround:** The FPSWA version 1.12 fixes this issue.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



### 47. Executing an rfi instruction that is located at the end of implemented physical memory can result in an unexpected unimplemented address fault

**Problem:** Due to this erratum, when the processor is in physical mode and an `rfi` instruction at the end of physically implemented memory is executed, the processor will take an unimplemented address fault regardless of the real target of the `rfi` (IIP).

**Implication:** On a platform that supports the full 50 bits of physical address, under the above conditions an unexpected unimplemented address (UIA) fault could occur and the result depends upon the implementation of the UIA fault handler. This issue has only been observed in a pre-silicon simulation environment.

**Workaround:** Do not place an `rfi` instruction at the end of implemented physical memory.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 48. IA-32: xchg instruction requires release semantics

**Problem:** The IA-32: `xchg` instruction can execute and write a value without it being explicitly ordered with respect to other IA-32 stores. The IA-32 memory model is strongly ordered and requires loads to have acquire (.acq) semantics and stores to have release (.rel) semantics to be executed in proper order. As a result of this requirement the `xchg` instruction requires the use of.acq and.rel semantics but only provides.acq semantics.

**Implication:** Due to this erratum, store operations may not be committed to memory in order with respect to IA-32 `xchg` operations.

**Workaround:** None at this time. PAL version 7.37 includes a fix for this issue.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 49. PAL MCA handler may not correctly set PSP.co bit

**Problem:** The PAL MCA handler may not set the continuable bit (PSP.co) for potentially recoverable errors.

**Implication:** If the PSP.co bit is not set on recoverable errors, the OS and/or application may terminate when they could have potentially recovered from the error.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 50. PAL_MC_ERROR_INFO may return incorrect PSP information

**Problem:** When the PAL MCA handler has detected a fatal condition or has requested a SAL_MC_RENDEZ procedure call, the PSP returned from the PAL_MC_ERROR_INFO procedure may not contain all error information.

**Implication:** If SAL_CHECK is using the PSP returned from the PAL_MC_ERROR_INFO procedure call, some error information maybe missing which could result in application termination or a system hang.

**Workaround:** SAL_CHECK should use the PSP data at PALE_CHECK hand off rather than from PAL_MC_ERROR_INFO.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 51. FPSWA trap may be missed

**Problem:** For Itanium 2 processor floating-point operations, when a *tiny*[1] result is computed (this usually corresponds to an underflow occurring), the processor should defer the computation to the FPSWA handler. In most cases, FPSWA will convert the result to a denormalized value that can be represented within the specified precision. However, for an extremely limited set of conditions, the processor fails to recognize this underflow and does not take the appropriate FPSWA trap.

**Implication:** Exposure to this issue occurs only under the following conditions:

1. Execution of one of the following instructions: `fma`, `fms`, `fnma`, `fpma`, `fpms`, `fpnma`.
2. The input operands for `fma`, `fms`, and `fnma` instructions (with or without.s or.d completers) must be capable of containing any combination of 64 bits in their significand, in register format. (If the significands of the operands are limited to less than 64 bits, the operation is not affected.)
3. The computed result is precisely $\pm 1.0 \times 2^{(Emin-1)}$ [2]. This is a necessary (but not sufficient) condition as only an extremely small subset of the possible input operand combinations that generate a result of $\pm 1.0 \times 2^{(Emin-1)}$ actually lead to a missed FPSWA trap. There must be a massive and specific cancellation generating the result prior to rounding to the destination precision.

For operations meeting these conditions, a small subset will not take the FPSWA trap. In these cases, the result ($\pm 1.0 \times 2^{(Emin-1)}$) will not be representable within the floating-point format specified. For example, assuming single precision mode, the result would be $\pm 1.0 \times 2^{-127}$. Normally, the FPSWA handler converts this result to a denormalized value in the form of
$\pm 0.1 \times 2^{-126}$ to fit within the single precision exponent format. Without this conversion the following impacts may be observed:

- For `fma`, `fms`, and `fnma` operations (with or without.s or.d completers) with FPSR.wre=0 [3], the result in the register file is numerically correct and may be used for subsequent floating-point operations without issue. However, storing this value to memory (using stfs, stfd or stfe as appropriate) will result in a correctly signed zero instead of $\pm 0.1 \times 2^{Emin}$. This is equivalent to what occurs for the "Flush-To-Zero" (FTZ)[4] mode of operation.

  It is possible to preserve the correct numerical result (that is, $1.0 \times 2^{-127}$ for the single precision example above) by using the `stf.spill` instruction for stores and the `ldf.fill` instruction for any subsequent loads.

- For register precision `fma`, `fms`, and `fnma` operations (with or without.s or.d completers) with FPSR.wre=1, the result should be $\pm 1.0 \times 2^{-65535}$. However, the result in the register file will be $\pm 1.0 \times 2^{-16382}$ in the form of a double-extended precision value.

- For parallel floating-point instructions (`fpma`, `fpms`, and `fpnma`), the result is stored in the register file as a correctly signed zero instead of $\pm 1.0 \times 2^{(Emin-1)}$. Parallel floating-point instructions are not used in any known compiled code.

---

1. A result is defined as tiny if it lies between $-2^{Emin}$ and $+2^{Emin}$ after rounding to the destination precision with unbounded exponent range. Reference the *Intel® Itanium® Architecture Software Developer's Manual* or IEEE Standard 754-1985 for Binary Floating-Point Arithmetic for any additional clarifications.
2. For single precision, Emin = –126; for double precision, Emin = –1022; for double-extended precision, Emin = –16382; for register format, Emin = –65534.
3. Reference the *Intel® Itanium® Architecture Software Developer's Manual* for Floating-point Status Register (FPSR) bit definitions.
4. FTZ mode causes tiny results to be truncated to the correctly signed zero.



**Workaround:** For the vast majority of floating-point usage models, no workaround is recommended. The issue is limited to an extremely small subset of possible floating-point operations with a typical impact of replacing a tiny value ($\pm$ 1.0 x $2^{(Emin-1)}$) with a correctly signed zero. Any error due to this issue is typically less, in absolute value, than the majority of rounding errors that normally occur for floating-point operations. For applications requiring a workaround, the following actions are required:

1. For `fma`, `fms`, and `fnma` operations (with or without.s or.d completers) with FPSR.wre=0, avoid input operands with 64-bit significands or use the `stf.spill` instruction for stores and the `ldf.fill` instruction for any subsequent loads.
2. Do not use register precision (FPSR.wre=1) for `fma`, `fms`, and `fnma` operations.
3. Do not use parallel floating-point operations (`fpma`, `fpms`, and `fpnma`).

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 52. WC evictions and semaphore operations combine to establish a potential live-lock condition

**Problem:** In the case that multiple processors are sharing memory space; when stores to WC memory are closely followed by semaphore operations to cacheable memory, the semaphore operations may block forward progress of the WC evictions. The semaphore will not be able to proceed until the WC stores are completed. As a result a live-lock condition is established between the WC evictions and the semaphore.

**Implication:** If the live-lock conditions are maintained, the system will eventually signal a BINIT. Other system activity or external interrupts may change availability of the system bus allowing the live-lock condition to be broken and the system will proceed as normal.

**Workaround:** None at this time. PAL version 7.37 includes a fix for this issue.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 53. The IA-32 cmpxchg8b instruction may not correctly set ZF flag

**Problem:** The IA-32 `cmpxchg8b` instruction should set the Zero Flag (ZF) flag to 1 and update memory when the compare operation is successful. However, if due to memory contention, the upper four bytes (bits 63:32) of the targeted memory are changed during execution of the instruction and the lower four bytes remain unchanged, the ZF flag may be incorrectly set to 1, even though the upper four bytes of the compare are not equal.

**Implication:** If this erratum occurs, two processors in a multiprocessor environment can end up owning the same memory locations when there should be autonomous ownership.

The failing scenario can only occur in a multiprocessor system where there is heavy contention for the targeted memory location. It also requires that another processor manages to update only the upper four bytes of the targeted memory location during a very small timing window just prior to execution of the compare.

This erratum only affects the `cmpxchg8b` form of the IA-32 `cmpxchg` instruction and has only been observed in a synthetic test environment.

**Workaround:** PAL version 7.40 includes a fix for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 54. PAL_TEST_PROC *status* return value

**Problem:** The PAL_TEST_PROC procedure returns *status* = –3 when the call has completed successfully and some self-test errors have occurred. Normally –3 would indicate that the PAL procedure itself has failed.

**Implication:** SAL firmware that assumes self-test errors will be reported with *status* = 0 may not function correctly.

**Workaround:** When PAL_TEST_PROC returns *status* = –3, SAL should check the *self-test_state* to obtain more information about the self-test error and report the error.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 55. Fault condition may generate incorrect address when using short format VHPT

**Problem:** A Debug Breakpoint or Protection Key fault may, under certain internal conditions, cause the physical address returned for a short format VHPT to not match the virtual address indicated by the VHPT entry.

The conditions under which this can occur are:

- The VHPT is enabled using the short format in a virtual addressing mode,
- Privilege level 0 access is available,
- Debug Breakpoint faulting is enabled (psr.db=1) and/or Protection Key Checking is enabled (psr.pk=1) and
- Certain cases of multiple TLB misses that result in multiple VHPT walks, where one of the VHPT walks is cancelled (because the faulting condition is removed) and then retried.

It is possible under these specific conditions that the short format data associated with the retried VHPT walk may be associated with another.

**Implication:** If this erratum were to occur, a Protection Key fault or an Instruction or Data Debug fault may cause a VHPT entry to be incorrect. This may result in an incorrect code sequence being executed and would leave the system in an indeterminate state.

With regard to Debug Breakpoint faulting, exposure is limited to development code environments only. In the case of Protection Key checking, there is no known exposure for all current operating systems as the conditions for this erratum are not met.

**Workaround:** This erratum affects only the short format VHPT, using the long format of the VHPT will avoid either of these faulting conditions. Additionally, in the case of Debug Breakpoint Faulting, prevent the DBR from ever matching any portion of the VHPT by checking the VHPT before allowing the DBR to be set.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 56. FPSWA version 1.12 may overwrite register fr12

**Problem:** The FPSWA version 1.12 may overwrite register fr12 when handling FPSWA faults caused by the `fma`, `fms` and `fnma` instructions consuming denormalized or unnormalized values. FPSWA should only use registers fr6-fr11.

**Implication:** Operating systems are required to save and restore fr6-fr11 when handling FPSWA faults. Any operating system that also saves and restores additional registers including fr12 is not susceptible to this issue. Depending on how an application uses fr12 and how the operating system preserves it, this erratum could lead to a number of different failure scenarios including incorrect data. The only known current exposure is with the Linux OS. This erratum is limited to FPSWA version 1.12.

**Workaround:** Upgrade to FPSWA version 1.18 or later which corrects the issue.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 57. Cache snoops disabled on BINIT#

**Problem:** After a BINIT# is signaled the processor will disable snoops to contain the propagation of any errors. The resulting MCA condition will cause the processor to enter the PAL MCA handler, which will invalidate the processor caches before the hand-off to SAL. The PAL MCA handler does not re-enable cache snoops before the hand-off to SAL.

**Implication:** This erratum only occurs after a BINIT event, thus any potential impact is limited to error handling after this fatal event. As a result of this issue cache coherency will not be maintained after a BINIT error. SAL code that runs uncacheable is unaffected. Cache coherency is restored after the processor is reset as part of the normal BINIT event handling.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



## 58. RFI to UIA using single step mode may enter ss trap

**Problem:** In single step mode, a single step trap may be incorrectly taken on an `rfi` instruction when the `rfi` attempts to address unimplemented memory.

**Implication:** The single step trap should not be taken on an `rfi` instruction. The result of this erratum would be an indication that the single step/`rfi` instruction was completed successfully before entering the unimplemented memory address (UIA) trap.

**Workaround:** Avoid taking an `rfi` to an UIA.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 59. On-Die Termination value does not meet specification

**Problem:** The value of the On-Die Termination (ODT) does not meet the specified range of 45 Ohms ±15% when measured at Vol. The actual value is 37 Ohms ±5% when measured at Vol. At output voltages above 0.6V, the ODT values are within the correct range.

**Implication:** The stronger value of ODT could result in a higher output low voltage (Vol) and reduced noise margins. Measurements on an Intel platform have not shown any noticeable increase in Vol and noise margins are within specified ranges.

**Workaround:** This erratum does not affect any system using on-board termination. No workaround is recommended for platforms using ODT in a 3-load configuration. ODT termination is not recommended for 5-load bus configurations, those should use on-board termination.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 60. Specific instruction combination may disrupt subsequent operation

**Problem:** A specific combination of memory and integer instructions may cause the result of a prior integer operation to be incorrect. The combination of instructions necessary for the failure is:

1. Four or more arithmetic and at least one additional operation executing concurrently, immediately followed by a subsequent integer operation that consumes data from the previous operation.
2. Particular data patterns are also required.
3. This erratum is more likely at higher temperatures and higher processor core speeds.

**Implication:** As a result of this erratum, an integer operation may consume incorrect data leading to unpredictable system behavior. In some instances, a fatal DTLB MCA or memory page fault may occur.

**Workaround:** Intel recommends implementing one of the following workarounds:

- Reduce the processor operating frequency to 800 MHz by adjusting the system bus ratio to 2:8. Consult the *Intel® Itanium® 2 Processor Hardware Developer's Manual* for complete information on setting the system bus ratio.
- Avoid use of the susceptible code sequence and/or add stops between affected instruction groups.

## 61. IFS register may be invalidated during MCA or INIT

**Problem:** If an interrupt service routine (ISR) is reading the interruption function state (IFS) control register when the processor detects an MCA or receives an INIT event, under certain internal timing conditions the destination register of the IFS read may indicate that the IFS is invalid.

To be exposed to this issue the processor must be in the proper context to read the IFS control register. This requires executing at privilege level 0, having interruption collection disabled (psr.ic=0), and the IFS register must be valid (ifs.v=1). Executing a `cover` instruction sets ifs.v=1. In addition MCAs and INITs must not be masked (psr.mc=0).

**Implication:** When the ISR issues a `rfi` instruction, the return value of current frame marker (CFM) may not be properly restored. The contents of the backing store application registers may not be correct in this situation. Indeterminate system operation can result if this erratum occurs.

**Workaround:** PAL version 7.59 for the Itanium 2 processor (up to 3 MB L3 cache) and PAL version 5.37 for the Itanium 2 processor (up to 6 MB L3 cache) contain a workaround that corrects the possible problem when reading the IFS control register. This workaround requires the OS to abide by some specific restrictions. All known current OS releases adhere to these restrictions. These restrictions are:

1. There are no branches within a small window of code after the IFS read. The length of this window is the shorter of either three bundles or two instruction groups.
2. A `cover` instruction must not be followed by a branch to a bundle within the window after the IFS read. The window is as defined in item #1.
3. All ISR code from the `cover` instruction to the earlier of either changing psr.ic to 1 or the `rfi` at the end of the ISR, must exist within the same contiguous region of physical memory.
4. A `bsw.1` instruction must not be used within the ISR after a `cover` instruction and prior to the IFS read. This applies only if the destination register of the `mov` from IFS is r29, r30, or r31. PAL version 7.71 for the Itanium 2 processor (up to 3 MB L3 cache) and PAL version 5.61 for the Itanium 2 processor (up to 6 MB L3 cache) removes the requirement for this restriction.
5. After an MCA or INIT event, if this workaround is unable to properly recover the IFS control register state, a fatal MCA will be signaled to prevent unpredictable machine behavior.
6. An additional restriction is that the *Dynamic Instruction Cache Prefetch* remain enabled (PAL_PROC_GET_FEATURES [46]=0) otherwise part of the workaround will be ineffective. This prefetch feature is enabled by default. This restriction has been removed in PAL version 7.77 and above for the Itanium 2 processor (up to 3 MB L3 cache) and PAL version 5.69 and above for the Itanium 2 processor (up to 6 MB L3 cache).

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 62. Unimplemented memory access may occur while handling an INIT or MCA event

**Problem:** This erratum involves possible incorrect behavior if an ISR or fault handler exists in physical memory near address zero. If such an ISR or fault handler is executing and a `cover` instruction has been executed (IPSR.ic=0, IFS.v=1) and then an INIT or MCA event occurs while the handler is within the address range of 0 to 0x20, the processor can incorrectly access unimplemented memory. This results in a second MCA generated by the incorrect PAL behavior and this MCA occurs while interruption collection (IPSR.ic=0) is disabled.

**Implication:** It is highly unusual that any part of an ISR or fault handler including the `cover` instruction would be located in the first few locations of physical memory. Current known OS releases are not affected. If this erratum were to occur, receiving nested MCAs is not a condition the OS expects to encounter. A system crash or fatal error event may occur.

**Workaround:** Do not locate ISR or fault handling code with a `cover` instruction within the physical address range of 0 to 0x20.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 63. JTAG Sample/Preload or EXTEST instruction usage

**Problem:** When using the JTAG Sample/Preload or EXTEST boundary scan instruction, all internal signals in the BSDL file must have their safe values loaded into the boundary scan serial data register when the JTAG state machine enters the update DR state. Failure to do so will result in putting the component into a non-operational test mode.

**Implication:** Failure to load the data register with safe values for all internal signals contained in the BSDL file may result in putting the part into a non-operational test mode.



**Workaround:** When loading the JTAG data register during the Sample/Preload instruction, or EXTEST instruction, load safe values contained for all internal signals contained in the BSDL files.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 64. CPU_CYCLES count includes data from halt states

**Problem:** The event monitor count for CPU_CYCLES accumulates the count of elapsed processor clock cycles even in a light halt state. The CPU_CYCLES counter is not expected to accumulate the count when the processor is in a light halt or powered down state.

**Implication:** The CPU_CYCLES performance monitoring event may report an incorrect count if the processor goes into a light halt state.

**Workaround:** PAL version 5.37 and above, for the Itanium 2 processor (up to 6 MB L3 cache) contain a fix for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 65. System bus signals can be driven while RESET# is asserted

**Problem:** Upon the first assertion of RESET# after PWRGOOD is asserted, the processor may drive some of the system bus signals. The processor should tristate all system bus signals within two bus clocks of the assertion of RESET#. Due to this erratum, the processor may not tristate all system bus signals within this two clock limit.

**Implication:** The system bus state during this initial time window with RESET# asserted cannot be determined. Since no processor execution takes place with RESET# asserted, this does not affect processor operation after the RESET# sequence has been completed.

**Workaround:** The state of the system bus signals during the initial RESET# sequence should be ignored.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 66. PSP.cr is always set to zero (0) at PALE_INIT hand off to SALE_ENTRY

**Problem:** When PALE_INIT completes the PAL handling of an initialization (INIT) event, status information is indicated in the Processor State Parameter (PSP) register at the hand off to SALE_ENTRY. After any INIT event, the state of PSP.cr (bit 20) will incorrectly be set to zero (0) which indicates that the control registers are not valid. This erratum only pertains to the state of the PSP.cr bit, the actual contents of all control registers after the INIT is correct and the control register information recorded by PALE_INIT in the min-state save area is also correct.

**Implication:** Based on the incorrect state of the PSP.cr bit, the control register information recorded in the min-state save area could be assumed to be invalid. In fact, the information is an accurate recording of the control register states at the time of the INIT event. Furthermore, the control registers are valid at the PALE_INIT to SALE_ENTRY hand off.

**Workaround:** The value of PSP.cr can be assumed to be one (1) (valid) after any INIT event.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 67. Incorrect Thermal Calibration Offset Byte value in the PIROM

**Problem:** The Thermal Calibration Offset Byte value in the PIROM was incorrectly programmed to eight (8). The correct value for the Thermal Calibration Offset Byte should be zero (0).

**Implication:** Systems using the Thermal Calibration Offset Byte value programmed in the PIROM may report inaccurate information for the following:

1. Temperature readings from the SMBus.
2. Upper and lower thresholds for THRMALERT#.

**Workaround:** Systems should use a value of 0 for the Thermal Calibration Offset Byte.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 68. Performance Monitoring Event counters may be incorrect after leaving a low-power state

**Problem:** On entry into the PAL_HALT_LIGHT procedure the performance monitoring counters that are expected to continue monitoring events in a low-power state will be frozen until the processor returns to full power.

**Implication:** As a result of this erratum, the Performance Monitoring Event counters noted in Section 10.3.11 of the *Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization* may be incorrect after leaving a low-power state.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 69. Instruction Breakpoint Register update may generate a false instruction debug fault

**Problem:** An incorrect instruction debug fault may be indicated on a write to the enable and mask bits in the Instruction Breakpoint Registers (IBR).

**Implication:** Code execution may fault on the false instruction debug fault generated by either the write into the IBR or on other instructions depending upon how the debug mask bits have been set. The IBR is only accessible in privilege level 0. OS software debug tools may or may not use this debug breakpoint feature.

**Workaround:** Disable Debug Breakpoint Faulting (Psr.db=0) before writing the enable and mask bits in the IBR and then re-enable Debug Breakpoint Faulting.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 70. Application fault may be missed on a br.ia instruction

**Problem:** An Illegal Operation Fault may not be indicated when executing the `br.ia` instruction and the BSPSTORE register is not equal to the BSP register.

**Implication:** An Illegal Operation Fault should be indicated if an unconditional branch (`br.ia`) into IA-32 application space is made without first issuing a Flush Register Stack (`flushrs`) instruction to ensure that BSP and BSPSTORE are equal and the register stack partitions are saved. As a result of this erratum it is possible that the IA-32 application code will begin execution before indicating a fault.

**Workaround:** Ensuring that a `flushrs` instruction is issued before executing the `br.ia` instruction, as required by the *Intel® Itanium® Architecture Software Developer's Manual*, will eliminate the exposure to this erratum.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 71. Machine check may not bring the system out of a low-power state

**Problem:** In the case that the processor has entered a low-power state and MCA checking is masked (PSR.mc=1) a machine check event may not bring the processor out of the low-power state.

**Implication:** The *Intel® Itanium® Architecture Software Developer's Manual*, Volume 2 (Document No. 245318) documents that the processor should return to the Normal state upon receipt of an unmasked external interrupt, machine check, Reset, PMI or INIT. As a result of this erratum a machine check event received in a low-power state while machine check aborts are being masked, will not be serviced until the system is returned to a normal operating state by any other wakeup event.

**Workaround:** Enable machine check abort checking (PSR.mc=0) before entering a low-power state.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 72. Machine check event received during PAL execution may have unexpected results

**Problem:** Depending on internal conditions, a machine check event (MCA) received during the execution of certain PAL procedures may have unexpected results.

**Implication:** During the execution of the following PAL procedures; PAL_CACHE_FLUSH, PAL_CACHE_INIT, PAL_CACHE_LINE_INIT, PAL_CACHE_READ, PAL_CACHE_WRITE, PAL_CAR_INIT, PAL_TEST_PROC and PAL_VM_TR_READ, if an MCA event is received



the PAL procedure may fail. Depending on when the MCA is received and the execution environment, the results may range from a PAL or system error to a processor hang. In most cases the procedure will execute correctly.

**Workaround:** Ensure that machine check abort checking is disabled (PSR.mc=1) before calling the PAL procedures noted above.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 73. Rendezvous may result in spin loop due to incorrect rendezvous address passed to SAL

**Problem:** When the PAL determines that an error has occurred which could cause a multiprocessor system to lose error containment, it must rendezvous the other processors in the system before proceeding with further processing of the machine check. This is accomplished by branching to SAL with a non-zero return vector address. It is then the responsibility of the SAL to rendezvous the other processors and return to PAL through this return address. It is possible for PAL to pass an incorrect return address to SAL during the hand off for processor Rendezvous.

**Implication:** The normal mode of operation during a rendezvous event is a blue screen, while the processors enter a spin loop. As a result of this erratum, the hand off to SAL may be fatal.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 74. Possible degradation in system performance when calling PAL_CACHE_FLUSH with int = 1 for certain cache memory types

**Problem:** When the PAL_CACHE_FLUSH procedure is called with int = 1, external interrupts will be polled periodically while the specified cache type(s) are being flushed. If an external interrupt is seen, this procedure will return and allow the caller to service the interrupt before all cache lines in the specified cache type are flushed. The problem is that when PAL_CACHE_FLUSH is called again to resume the flush operation from where it was interrupted, PAL attempts to start the flush operation over again rather than continuing from the point of interruption. This erratum affects *cache_types* 1, 2, and 3 as described in the The *Intel® Itanium® Architecture Software Developer's Manual*, Volume 2 (Document No. 245318).

**Implication:** If additional interrupts continue to occur before the completion of the PAL_CACHE_FLUSH, the procedure may never complete. This may result in degraded system performance due to one processor not being available or appearing to be stalled. This issue has only been observed in a validation test environment.

**Workaround:** Do not call the PAL_CACHE_FLUSH procedure with int = 1 and *cache_type* = 1, 2 or 3.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 75. Memory read current transaction may fail to observe a st, ld.bias or lfetch.excl

**Problem:** A memory read current transaction allows a chipset to access a coherent copy of a cache line in a caching agent without affecting the cache line state in the caching agent. This transaction avoids later cache misses and subsequent transactions by the cache agent to again cache the line.

The erratum requires the following code sequence:

1. Given two addresses X and Y, which would map to two different L2 cache lines:
   a. A memory read current (same cache line as X) must occur coincident to the sequence: load(X)... load (same cache line as X)... store (same cache line as X);
   ***or***
   b. A memory read current (same cache line as X) must occur coincident to the sequence: load(X)... semaphore (Y)... store (same cache line as X);
   ***or***
   c. Either of the above where store(X) is replaced with an ld.bias(X) or an lfetch.excl(X).

2. First `load(X)` need not be cached but has to fill the L2 to an E-state.

If systems utilize the memory read current transaction and execute the above code sequence, and specific internal micro-architectural timings are met, the cache line may be updated to an incorrect state by the processor.

**Implication:** Usage models are not known to exist where the *st*, *`ld.bias`* or *`lfetch.excl`* to a cache line (X) at or near the time of a memory read current transaction targeting cache line (X). If the conditions as described are met, a future external access to the memory contained in cache line (X) will not receive the expected *hitm* snoop response from the processor. Internal accesses will miss and be issued to the system interface.

**Workaround:** Memory read current transactions should not be used in situations where the above conditions are met.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 76. BINIT taken on 2x ECC and hard-fail errors with BINIT event signaling disabled

**Problem:** A Bus Initialization (BINIT) event may still be signaled after a multiple-bit ECC or hard-fail error, even if BINIT event signaling/checking is disabled.

**Implication:** Multiple bit ECC errors, PTC and IPI operations that experience transactions errors may normally signal a Machine check that result in a BINIT response. However, when the BINIT response is disabled a BINIT is not expected. As a result of this erratum a BINIT will still be signaled for these types of errors even with the BINIT response disabled.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 77. Recoverable L3 cache tag ECC error may raise overflow error when CMCI are promoted to MCA

**Problem:** In the case that CMCIs are promoted to MCA, certain internal conditions combine with an L3 cache tag ECC error to indicate an overflow error and signal a fatal MCA.

**Implication:** An L3 cache tag ECC error is normally a recoverable CMCI but when CMCIs are being promoted to MCA, the error is promoted as a fatal MCA event instead of being firmware corrected. The fatal MCA is indicated if the cache line tags are snooped after the ECC error is flagged but before the MCA is taken.

**Workaround:** A workaround is under investigation.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 78. L2 cache line with poison data results in unexpected fatal MCA

**Problem:** An L2 cache line with latent 2x ECC or poisoned data that is snooped before being consumed may incorrectly signal a fatal MCA.

**Implication:** An L2 cache line with a 2x ECC or an error that results in a cache line being poisoned should indicate a CMCI unless the data is consumed by a processor. A subsequent snoo.p hit to the poisoned cache line may cause the errant line to be flagged as an error twice, which would result in a machine check overflow and a fatal MCA being taken rather than a CMCI. This erratum does not apply to consumed poisoned data

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 79. XPN time-out with BINIT response disabled may cause system hang

**Problem:** In the case where the BINIT response to a processor internal time-out response is disabled, a second XPN time-out error may result in a system hang.

**Implication:** If an XPN time-out occurs such that a BINIT should be taken but is not due to the fact that the BINIT on an internal time-out response has been suppressed. A second XPN time-out error may result in the system hanging because the time-out counter was not reset after the first internal time-out.

**Workaround:** Do not suppress the BINIT response to a processor internal time-out.

**Status:** For the steppings effected, see the *Summary Table of Changes*.



### 80. BINIT may be taken after a UC single byte access to ignored/reserved area of the Processor Interrupt Block

**Problem:** A Bus INITialization (BINIT#) event may be signaled after an uncacheable (UC) single byte access to any ignored/reserved area in the upper half of the Processor Interrupt Block.

**Implication:** Unsupported accesses result in undefined behavior of the processor, hence the BINIT# response is taken to re-establish a consistent execution environment. In other cases the unsupported access can be ignored. Single byte UC access to the ignored or reserved areas of the IPI block should be ignored but as a result of this erratum a BINIT# is signaled.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 81. Recoverable CMCI may combine with an L3 MCA error to cause fatal overflow error

**Problem:** In the case where a recoverable L3 cache or system bus error flags a Correctable Machine Check Interrupt (CMCI) and is followed by specific MCA events, the overflow bit may be set and result in a fatal error. The specific MCA events are a L3 cache, system hard-fail, local BINIT# or a non-coherent UC/WC memory access that receives a HITM response.

**Implication:** As a result of this erratum a CMCI or MCA event that is normally recoverable, if supported by the OS, may set the overflow bit and signal a global BINIT#.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 82. BERR may be indicated when the PAL MCA routine invalidates L2 cache lines

**Problem:** A Bus ERRor (BERR#) may be signaled when a read hit occurs to the same L2 cache line that a PAL MCA routine is in process of invalidating.

**Implication:** As a result of this erratum a BERR# may be signaled after a hard-fail error, if a read hits a cache line while the line is being invalidated via the MESI protocol tags but before the cache line ECC has been updated.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 83. Pending RSE interrupt during the PAL PMI handler PAL PMI flow may result in a system hang

**Problem:** A system hang may be the result of a case that we have a pending RSE interruption that occurs during the execution of the PAL PMI handler.flow.

**Implication:** Depending on the execution of the PAL PMI flow and a pending RSE interruption, the result may be unsuccessful handling of the PAL PMI handler which would lead to a system hang.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 84. An INIT signaled during the PAL PMI flow while a PAL PMI flow RFI is being serviced may result in a system hang

**Problem:** If an MCA/INIT is signaled during the execution of the PAL PMI handler when an `rfi` is in the instruction pipeline but not yet executed, there exists a window of exposure in which a system may hang as the `rfi` is aborted before returning from the MCA/INIT procedure.

**Implication:** There is a small window of exposure where If the `rfi` can be in the instruction pipeline and an MCA/INIT is taken, where it aborts the `rfi` before the `rfi` has been executed. If these above mentioned conditions are met the result may be a system may hang.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 85. PMI serviced during the execution of PAL_MCMA_ERROR_INFO procedure may result in unpredictable processor behavior

**Problem:** If a PMI is taken during the execution of the PAL_MC_ERROR_INFO procedure, the branch return information stored by the PAL call may be lost. As a result, the behavior of the processor is not guaranteed upon its return from the PMI handler.

**Implication:** PAL_MCMA_ERROR_INFO may not complete successfully and the processor behavior is unpredictable.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 86. Data-poisoning bits not included in PAL_MC_ERROR_INFO cache_check and bus_check structures

**Problem:** In the *Intel® Itanium® Architecture Software Developer's Manual Specification Update* machine check architecture extensions were added for supporting data-poisoning events. These extensions will help in supporting different data-poisoning handling policies. Current Itanium 2 processors do not implement the dp bit in the cache_check and bus_check structures in PAL_MC_ERROR_INFO.

**Implication:** When parsing error logs, the OS cannot distinguish between some hardware generated corrected events versus data-poisoning events.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 87. PAL_PREFETCH_VISIBILITY call not implemented

**Problem:** Calling PAL_PREFETCH_VISIBILITY with trans_type argument of 1 returns Invalid Argument.

**Implication:** PAL_PREFETCH_VISIBILITY does not support physical addressing attribute transitions.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 88. INIT# signal not recognized properly

**Problem:** The INIT# signal triggers an unmasked interrupt to the processor. When operating at odd bus-to-core frequency ratios, the assertion of the INIT# pin may not always be recognized by the processor, preventing the processor from taking the interrupt.

**Implication:** Due to internal timing and electrical conditions, it is possible that the processor may not recognize the INIT# signal when odd bus ratios (that is, 2:9, 2:11, and so forth) are being used. This erratum is intermittent in nature and could result in the system missing an INIT# assertion.

***Note:*** This erratum does not impact the use of the INIT# pin for power-on configuration during reset, nor does it affect other system interrupts.

**Workaround:** One of the following two workarounds can be implemented:

- Either a system bus-based interrupt transaction or the Platform Management Interrupt (PMI)# input can be used to implement the same functionality. In this case the PAL_PMI code flow will handoff control to SAL_PMI. The SAL_PMI code can check the status of the INIT# signal and if INIT# has been asserted, the SAL code flow can call SAL_INIT.
- Early in the SAL_INIT code, send an INIT IPI to all other processors in the domain. The following issues should be considered to build a more intelligent SAL_INIT implementation:
  - — Do not call PAL_MC_RESUME during INIT IPI handling.

— If there is any “timeout” mechanism in the INIT handling flow, that value may need to be increased to reflect the fact that some processors will see INIT#/IPI earlier than others.

— INIT IPIs could be sent only to other processors that have not yet seen the INIT#, this would be necessary in the case where the SAL/OS INIT code unmasks MCAs (PSR.mc=0). However, it is typical that MCAs are masked (PSR.mc=1) on the first INIT, so multiple INITs received by a given processor should not cause a problem for INIT handling flow as further INITs should be pended but not recognized.

— Consider the processor and ratios in effect in order to determine the necessity of this workaround.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 89. Cache lines with ECC errors may not be invalidated

**Problem:** In some instances, cache lines with single-bit errors may not be invalidated as expected.

**Implication:** Multiple CMCIs may be seen for the same single-bit error as it will remain in the L2 or L3 cache until flushed by regular system execution. The single-bit errors are automatically corrected when data is requested.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 90. Interrupts are enabled when exiting from a halt state

**Problem:** When exiting from PAL_HALT, PAL_HALT_LIGHT, or PAL_HALT_LIGHT_SPECIAL, PSR.ic is incorrectly set.

**Implication:** Interrupts are enabled after the processor wakes from the halt state.

**Workaround:** Disable interrupt collection within the SAL code flow.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 91. PAL_PREFETCH_VISIBILITY call may result in a system hang

**Problem:** Calling PAL_PREFETCH_VISIBILITY with trans_type = 1 could result in PAL entering a spin loop.

**Implication:** PAL_PREFETCH_VISIBILITY does not support physical addressing attribute transitions.

**Workaround:** Do not call PAL_PREFETCH_VISIBILITY with trans_type = 1.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 92. Corrected ECC error may not generate CMCI

**Problem:** A hardware corrected error may not generate a CMCI when an IPI or PTC transaction is in progress.

**Implication:** In the case of a 1xECC error on an IPI or PTC transaction, a hardware corrected CMCI may not be signaled to the operating system even if CMCI signaling for hardware corrected errors is enabled. It is important to note that the 1xECC error is detected and corrected by the processor and has no impact to the executing processes.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 93. PAL_CACHE_FLUSH procedure may not flush and invalidate all L2 cache lines

**Problem:** In the case that a PAL_CACHE_FLUSH procedure is called to flush and invalidate the L2 cache lines, cache lines that are in the exclusive or shared state may not be invalidated.

**Implication:** As a result of this erratum, the PAL_CACHE_FLUSH procedure may not be successful in invalidating the exclusive or shared cache lines. However, all modified lines are written to memory and with the memory copy being valid for exclusive and shared state, all memory is up-to-date at end of routine.

**Workaround:** Replace the PAL_CACHE_FLUSH procedure call with the "`fc`" instruction to cover the address range to be flushed.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 94. Performance counters may include data from low power states

**Problem:** The following list includes a number of processor performance counters that may continue to accumulate event counts in a low power state.

- BACK_END_BUBBLE.ALL
- BACK_END_BUBBLE.FE
- FE_BUBBLE.ALL
- FE_BUBBLE.BUBBLE
- FE_BUBBLE.GROUP1
- FE_BUBBLE.ALLBUT_IBFULL
- FE_LOST_BW.ALL
- FE_LOST_BW.BUBBLE
- BE_LOST_BW_DUE_TO_FE.ALL
- BE_LOST_BW_DUE_TO_FE.BUBBLE
- IDEAL_BE_LOST_BW_DUE_TO_FE.ALL
- IDEAL_BE_LOST_BW_DUE_TO_FE.BUBBLE

**Implication:** These performance counters are not expected to continue to accumulate data in a low power state. As a result of this erratum the count for these events may be inaccurate after leaving a low power state.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 95. MCA due to an XPN timeout may result in a spin loop

**Problem:** If MCAs have been enabled to occur at the halfway count of an XPN timeout, PAL may enter a spin loop.

**Implication:** Instead of passing the MCA up to SAL, PAL incorrectly enters a spin loop.

**Workaround:** Disable the MCA at the halfway count through PAL_PROC_SET_FEATURES.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## 96. BINIT# may not be asserted for exactly two cycles

**Problem:** As stated in the *RS - Itanium® 2-Based Platform Compatible Processors System Bus Specification*, if an agent samples BINIT# asserted on clock N, and it asserts BINIT# for the first time in cycle N, then the agent must keep BINIT# asserted for exactly two cycles. Currently all *Itanium® 2* processors assert BINIT# for one cycle in the scenario described above.

**Implication:** The agents on the system bus have one clock cycle to sample asserted BINIT#. Actions taken upon sampling the asserted BINIT# remain unchanged and are listed in the *RS - Itanium® 2-Based Platform Compatible Processors System Bus Specification*.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



### 97. Memory read current transaction may fail to observe a st or lead to a system hang

**Problem:** A memory read current transaction allows a chipset to access a coherent copy of a cache line in a caching agent without affecting the cache line state in the caching agent. This transaction avoids later cache misses and subsequent transactions by the cache agent to again cache the line.

The erratum requires the following code sequence:

1. Given an addresses X which maps to a L2 cache line and an address Y which maps to a cache line that belongs to the same set as X at L2.
2. A memory read current (same cache line as X) must occur coincident to the sequence. load(X)... store (same cache line as X)... load (same cache line as Y); If systems utilize the memory read current transaction and execute the above code sequence, and specific internal micro-architectural timings are met, subsequent transactions may not return the correct data and may lead to a system hang.

**Implication:** Usage models are not known to exist where the st to a cache line (X) at or near the time of a memory read current transaction targeting cache line (X). If the conditions as described are met, even though the st is correctly posted to the cache line by the processor, incorrect data is returned for subsequent system interface accesses to a different cache line. Another possible impact of the erratum is a system hang due to erroneous assertion of the HIT# and HITM# snoop signals for accesses to the cache line.

**Workaround:** Memory read current transactions should not be used in situations where the above conditions are met.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 98. PAL_VM_TR_READ will return an incorrect page size for DTR reads

**Problem:** When calling PAL_VM_TR_READ with tr_type = 1 (DTR), the return ps field will hold an incorrect value.

**Implication:** The value returned by the PAL_VM_TR_READ procedure cannot be relied upon for informational or architectural implementations.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 99. Incorrect EID and ID information passed by PAL

**Problem:** Itanium 2 processor PAL, incorrectly reports the EID and ID mask bits in GR33[31:16] instead of GR33[47:32].

**Implication:** EID and ID bits cannot be relied upon for a correct representation of the programability of the LID register.

**Workaround:** In the case that the information about the programmable bits of the LID register is required by SAL, the following steps should be taken:

1. Write 1's to the LID register.
2. Follow the write with a read from the LID register.
3. Bit positions with a read back value of 1 are programmable whereas bit positions with a read back value of 0 are read-only.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 100. Interruption of PAL calls by a PMI or INIT

**Problem:** In the case where a PMI or INIT interrupts a PAL procedure and the handler makes a PAL call, the processor may take a general exception fault.

**Implication:** Normal operation of the processor is not guaranteed in the above mentioned scenario. It must be noted that, for this issue to occur, the PAL call made in the interruption handler must alter the machine state used by the interrupted PAL procedure.

**Workaround:** None at this time.

**Status:** Initial fix for this erratum is found in PAL versions 7.78, 7.79, 5.72, 5.73, 2.10, 2.14, and 2.15. An extension has been added in future PAL versions to handle the corner case when the register frame is incomplete.

### 101. External interrupt polling and PAL_CACHE_FLUSH

**Problem:** If PAL_CACHE_FLUSH is called with external interrupt polling enabled (int =1) and an interrupt occurs during the PAL procedure, the returned progress indicator may be invalid. It must be noted that this issue only affects an Itanium 2 processor with a cache size smaller than 9MB.

**Implication:** Subsequent calls to PAL_CACHE_FLUSH that use an incorrect progress indicator will return an invalid argument.

**Workaround:** Call PAL_CACHE_FLUSH without enabling interrupt polling (int=0).

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 102. PAL_MC_ERROR_INFO call could invalidate incorrect cache line entry

**Problem:** When an L3 cache 1xECC error is detected by the processor and PAL_MC_ERROR_INFO is called, the processor may perform an L3 cache line invalidate operation. There exists a small window where the cache line may be used, and the invalidate operation will target the incorrect entry. PAL must be running in a cacheable mode for this to occur.

**Implication:** Unpredictable system behavior.

**Workaround:** Call PAL_MC_ERROR_INFO in uncacheable mode.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 103. L3 cache tag error and pending cache line replacement transactions may result in system livelock

**Problem:** An L3-tag 1xECC error in combination with several pending cache line replacement (BCR) transactions, internal timing conditions and a single snoop can establish a potential livelock condition.

**Implication:** This erratum may result in a system hang, however the livelock condition may be broken by any additional snoops, instructions entering the pipeline or completion of any of the BCR transactions that were required to establish the livelock condition.

**Workaround:** Cache line replacement transactions should not be used or L3 in-line correction mode can be enabled (see the *Itanium® 2-Based Platform Compatible Processors Firmware Guide* for details) in order to avoid this potential livelock condition. PAL version 2.10 provides an alternative workaround through PAL_PROC_SET_FEATURES feature_set 0x11 bit 5.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 104. SALE_ENTRY may see unexpected modified cache line during system cold boot

**Problem:** During a system cold boot, PAL firmware may unexpectedly leave a modified cache line entry in L3 cache on hand-off to SALE_ENTRY. PAL is expected to not leave any entries in cache for the hand-off to SAL.

**Implication:** The cache line is written by the PAL initialization process and is targeted to memory. As a result a memory error may be indicated during the cold boot process. Intel continues to recommend that any memory errors received before the memory levelization and initialization process has been completed by SAL, should be ignored.

**Workaround:** SALE_ENTRY should be ignoring memory errors until memory initialization is complete. A future PAL version will contain a fix for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 105. Lower priority error flagged on illegal write to GR r0

**Problem:** A Reserved Register/Field Fault may be incorrectly taken in place of an Illegal Operation Fault when an alloc instruction targets General Register (GR) r0.

**Implication:** The write to GR r0 is an illegal operation and should return an Illegal Operation Fault. However, if the `alloc` instruction is also trying to change the size of CFM.sor with the

register rename base registers (CFM.rrb.gr, CFM.rrb.fr, CFM.rrb.pr) not set to zero, the lower priority Reserved Register/Field Fault is flagged instead.

**Workaround:** A `clrrrb` instruction should be issued before the `alloc` attempts to change the size of the CFM.sor.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 106. PAL_TEST_PROC L3 cache replacement test may return invalid response

**Problem:** When calling PAL_TEST_PROC with the L3 cache replacement test enabled, the procedure may return an invalid performance restricted response on the *Intel® Itanium® 2 Processor 1.40 GHz with 1.5 MB L3 Cache* and *Low Voltage Intel® Itanium® 2 Processor 1.0 GHz with 1.5 MB L3 Cache.*

**Implication:** The PAL_TEST_PROC L3 cache replacement self test procedure may incorrectly return a performance restricted response but there is no actual degradation.

**Workaround:** Disable the L3 cache replacement late self test.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 107. PAL_CAR_INIT may not clear all cache lines

**Problem:** The PAL_CAR_INIT procedure may not clear all cache lines to null upon exit of the PAL procedure.

**Implication:** Upon exit of the PAL_CAR_INIT call, random data may be unexpectedly left in some cache lines.

**Workaround:** SAL code should clear the data portion of the cache.

**Status:** For the steppings affected, see the *Summary Table of Changes*

### 108. PSR.IC may not be restored properly on exit from a PAL call

**Problem:** If a PAL call is made with both PSR.IC and PSR.MC set, PAL may not restore the PSR.IC bit properly on exit.

**Implication:** Interrupt collection may be unexpectedly turned off after a PAL call is made. This may result in a system hang after a fault is taken.

**Workaround:** None available at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 109. Performance counters may not be correctly restored upon exit of the LIGHT HALT state

**Problem:** A call to PAL_HALT_LIGHT will place the processor in the LIGHT HALT state. In this state, select performance counters should remain frozen. In certain instances, upon exit of the halt state the overflow bit of the performance counters may be incorrectly set.

**Implication:** Incorrect performance monitoring values could be used after the exit of PAL_HALT_LIGHT.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 110. Single-bit errors in the tag and data portion of cache lines in the "I" state in the L2 or L3 levels of cache may not be flushed

**Problem:** Single-bit errors in the tag portion of lines in the "I" state in the L2 or L3 levels of cache may not be flushed by calls to PAL_MC_ERROR_INFO. Any access to lines within the same cache set will signal the error with a CMCI.

**Implication:** Single-bit errors will remain in I-lines until evicted by normal system execution.

**Workaround:** SAL can cleanse the entire cache with a call to PAL_CACHE_FLUSH followed by PAL_CACHE_INIT

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 111. Un-initialized word lines at processor boot could result in an incorrect branch address

**Problem:** A small percentage of processors may intermittently boot up with un-initialized word lines on the branch register file. These word lines are duplicated across four read ports. The first access of a branch register on a given port, before all of the word lines on the same port have been initialized, could result in an incorrect branch address being calculated.

These branch register word lines are initialized by any read access, including predicated code that is not used, and are subsequently not exposed.

**Implication:** The most common symptom is for the processor to fail to boot. However subsequent branches could use an incorrect address, leading to an illegal address fault or unpredictable system behavior.

**Workaround:** A power cycle re-boot can be used to work around the non-boot implication. For full port initialization, SAL or PAL Itanium® 2 Processor (up to 9 MB L3 cache) versions 2.20 and beyond will run a series of instructions to initialize all word lines and will eliminate the exposure of incorrect branches post-boot.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 112. Unexpected MCA on a fill to a line with parity errors

**Problem:** In the case that a fill hits a line with parity errors and certain microarchitectural conditions are met, the processor may report an unexpected MCA.

**Implication:** The processor may report two MCA's when in reality only one should have been reported for the parity errors on the line. It must be noted that the coherency is maintained for the processor.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 113. Performance associated with an epc instruction

**Problem:** The `epc` (enter privileged code) instruction increases the privilege level without causing an interruption or a control flow transfer. In the case that an epc instruction is in the same bundle and follows a branch instruction, a pipeline flush may be observed even if the branch instruction is correctly predicted.

**Implication:** An unexpected performance penalty may be observed if the above conditions are met.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 114. Branch bit, mispredict bit and slot index of branch instruction

**Problem:** The branch trace buffer provides information about the outcome of the most recent Itanium branch instructions. For every qualified Itanium branch instruction, the source bundle address and slot number are written to the branch trace buffer. Due to an issue, bits [4:0] of the branch trace buffer may not be captured correctly.

**Problem:** As a result the branch bit, mispredict bit and the slot index of the branch instruction in the bundle may not be accurate

**Workaround:** None at this time.



**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 115. Lower priority error flagged on illegal write to GR r0

**Problem:** A Reserve Register/Field Fault may be incorrectly taken in place of an Illegal Operation Fault when an alloc instruction targets General Register (GR) r0.

**Implication:** The write to GR r0 is an illegal operation and should return an Illegal Operation Fault. However, if the `alloc` instruction is also trying to change the size of CFM.sor with the register rename base registers (CFM.rrb.gr, CFM.rrb.fr, CFM.rrb.pr) not set to zero, the lower priority Reserved Register/Field Fault is flagged instead

**Workaround:** A `clrrrb` instruction should be issued before the `alloc` attempts to change the size of the CFM.sor.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 116. ptc.e instructions may purge resources of the other logical processor executing on the same core

**Problem:** One or more translation entries are purged from a single processors instruction and data translation cache by a `ptc.e` instruction and it should not propagate to other processors in the system. However, in a multi-core, hyper-threaded environment, `ptc.e` may purge one or more translation entries from the other logical processor running on the same core.

**Implication:** It must be noted that the usage of `ptc.e` is synchronized and currently existing software should not see any performance impact as a result of this issue.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 117. MPE_SCB_LIVE_REQ counts for disabled cores

**Problem:** MPE_SCB_LIVE_REQ counts even if the “either” unmask of a single core SKU is set.

**Implication:** Single core SKU may get spurious results due to this issue and the results should not be relied upon.

**Workaround:** Set the umask to “self” instead of “either” for single core SKU’s.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 118. move to bspstore requires unexpected serialization

**Problem:** When a br.ia is executed with bspstore != bsp an illegal operation fault should be raised and when psr.di is set, the machine should assert disabled instruction set transition fault. In the case that the above mentioned conditions occur in secession, a disabled instruction set transition fault is detected instead of a illegal operation fault (illegal operation fault has higher priority).

**Implication:** There is no architectural serialization required after a move to bspstore (typical of application registers) but as a result of this issue move to bspstore requires unexpected serialization.

**Workaround:** It is recommended that a srlz.i be executed prior to a br.ia.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 119. System behavior as a result of nested BINIT’s

**Problem:** There exists a possibility of system hang in the case that an MCA or a BINIT is being serviced by the processor and a 2-bit ECC error occurs in the L2 data cache tag-array of the processor.

**Implication:** Nested BINIT’s of the type described above are not guaranteed to be logged/handled correctly and the behavior of the machine is not guaranteed in the above mentioned condition.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 120. Instruction Pointer-Event Address Register (IP-EAR) may not behave as specified

**Problem:** Instruction Pointer-Event Address Register (IP-EAR), when enabled in a MT environment, may not behave as specified during a logical processor switch. Depending on the status of the incoming logical processor's IP-EAR, it is possible that an expected entry may get dropped in the outgoing logical processor's IP-EAR or an incorrect entry may get logged into the incoming logical processor's IP-EAR.

**Implication:** IP-EAR may not behave as specified.

**Workaround:** Disable the IP-EAR in a MT environment. Please refer to Chapter 3 of the *Update to the Intel® Itanium® 2 Processor Reference Manual For Software Development and Optimization* (Document #18612).

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 121. Performance Monitor Data (PMD) registers 10-15 usage

**Problem:** The Performance Monitor Data (PMD) registers 10-15 cannot be used to enable the overflow and freeze capabilities when HT technology is enabled.

**Implication:** Do not use the PMD registers 10-15 for overflow and freeze capabilities.

**Workaround:** Use the PMD registers 4-9 to enable the overflow and freeze capabilities.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 122. Wrong address generated for L3 data 1x and 2x ECC errors

**Problem:** This issue occurs when two cache lines, A and B, both exist modified in the L3 cache and line A has an error in the data array. If line A is snooped just before line B is snooped with the snoop confirms for both lines being close together, then the error may be logged for line B instead of line A.

**Implication:** The wrong target address may be reported by PAL_MC_ERROR_INFO for L3 data 1x and 2x ECC errors generated from snoop events.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 123. Illegal opcodes may not raise the expected operation fault

**Problem:** In some rare instances, when running with HT technology enabled, some illegal opcodes may not raise the expected illegal operation fault when executed.

**Implication:** For application code, this may result in a privileged operation fault instead of an illegal operation fault.

**Workaround:** None at this time.

**Status:** For the steppings effected, see the *Summary Table of Changes*.

### 124. Logical Processor Migration (LPM) is not working as expected

**Problem:** Logical Processor Migration (LPM) is included in PAL Rev 5.10 with minimal testing.

**Implication:** The following issues may arise when running LPM including:

- Pending register stack engine faults may not be handled correctly during LPM procedures, which may result in a system hang.
- The register stack engine may not be restored correctly, which may result in a general exception fault.
- TLBs may not be restored properly, which may result in a system hang.
- Region registers may not be saved or restored properly.

**Workaround:** None.

**Status:** For the steppings affected, see the *Summary Table of Changes.*



### 125. ALAT test is unavailable

**Problem:** PAL_TEST_INFO returns a value of 1 in bit 12 of st_control indicating that the ALAT test is unavailable.

**Implication:** The ALAT test is unavailable. Use the configuration defined by st_control that is provided by PAL_TEST_INFO.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 126. Internal processor timeout (XPN) events are not signaled

**Problem:** When a processor XPN time out occurs the half-timeout MCA (if enabled) and full timeout BINIT# assertion (if enabled) will not occur.

**Implication:** Half-timeout MCA (if enabled) and full timeout BINIT# assertion (if enabled) will not occur.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 127. PAL incorrectly interpreting updates to the Virtual Processor Descriptor

**Problem:** Updates to the Virtual Processor Descriptor are not being interpreted by PAL correctly leading to the Set System Mask(ssm) instructions not delivering the expected fault to the expected virtual external interrupt vector of the Virtual Memory Manager.

**Implication:** This issue results in the fault not being handled correctly and may result in unexpected behavior and when using Virtualization Technology.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the Summary Table of Changes.

### 128. PAL based IA-32 execution may result in unpredictable behavior

**Problem:** When software or hardware unexpectedly evicts a translation cache entry that is utilized in PAL based IA-32 execution, a system hang may result.

**Implication:** System hangs and/or unpredictable behavior may be observed due to this issue.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 129. Two MCAs issued due to active logical processor being switched

**Problem:** If the active logical processor is switched just after an error event; the MCA will be pended to the wrong logical processor, leading to two MCAs being issued.This issue only occurs with hyper-threading enabled.

**Implication:** This may result in a recoverable MCA turning into a fatal MCA and may cause a system reset.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 130. PAL_VP_SAVE and PAL_VP_RESTORE procedures not working as expected

**Problem:** PAL_VP_SAVE and PAL_VP_RESTORE procedures do not save and restore the implementation-specific state of PAL_PROC_GET/SET_FEATURES and PAL_GET/SET_PSTATE as requested by the pal_proc_vector parameter.

**Implication:** Setting the pal_proc_vector bits have no effect.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 131. PAL_VP_REGISTER procedure not working as expected

**Problem:** The PAL_VP_REGISTER procedure does not correctly populate the general exception and virtualization fault vectors with the configuration information needed for correct handling.

**Implication:** Calling PAL_VP_REGISTER will result in incorrect handling of virtualized instructions and may result in a system hang.

**Workaround:** If the Virtual Machine Monitor (VMM) needs to move the (IVT) Interrupt Vector Table location; the VMM needs to bring down the virtual machines, perform the move, and bring the virtual machines back up.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 132. Write access to a cache line with an uncorrectable error results in a MCA instead of a CMCI

**Problem:** Any write access to a cache line with an uncorrectable error may cause a recoverable MCA instead of the expected CMCI.

**Implication:** This error condition may result in an application crash or a system reboot.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 133. PAL_CACHE_SHARED_INFO not working as expected

**Problem:** PAL_CACHE_SHARED_INFO may incorrectly report that cache levels are being shared between logical processors even when hyper-threading is disabled.

**Implication:** None.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 134. PAL_MC_ERROR_INJECT in the cache error consumption mode may not work as expected

**Problem:** Calling PAL_MC_ERROR_INJECT in the cache error consumption mode may result in a general exception fault.

**Implication:** This issue may lead to a system reset.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 135. TLB consumption mode in PAL_MC_ERROR_INJECT uses the incorrect address

**Problem:** The TLB consumption mode in PAL_MC_ERROR_INJECT uses the incorrect address due to PAL miscalculating the consumption address. PAL attempts to access the TLB entry in which an error has been seeded, and an incorrect virtual address is used.

**Implication:** This can result in either a data or instruction TLB fault if the mis-calculated address does not reside in the TLB or in the case where a matching address is located in the instruction TLB, this could lead to undefined system behavior.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 136. PAL correction of any L2D or L3 correctable error on a cache line may flush that line

**Problem:** The PAL correction of any L2D or L3 correctable error on a cache line may inadvertently flush that line. The PAL correction of any L2D or L3 correctable cache error may trigger the MESI = Invalid behavior as described below. The PAL correction of any correctable L3 cache error may trigger the MESI = Disabled behavior as described below.

**Implication:** The incorrect flushing of the line can lead to the following situations:

MESI = Invalid - An unimplemented address is flushed out of the processor. System behavior is dependent upon chipset implementation and may lead to a system reset.

OR

MESI = Disabled - Loss of Intel® Cache Safe Technology tracking data may occur. This may result in recurring transient events requiring one more occurrence than expected prior to Intel® Cache Safe Technology removal.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 137. PAL improperly decodes the instruction in response to a virtualization fault

**Problem:** When running under Virtual Machine Monitor software, PAL improperly decodes the instruction in response to a virtualization fault. PAL mistakenly decodes the mov.itc imm instruction to a rfi instruction.

**Implication:** This may lead to the virtual processor having undefined behavior, which may result in a hang of the guest OS.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the Summary Table of Changes.

### 138. BINIT# assertion may result in a system hang.

**Problem:** In rare cases, a BINIT# assertion occurring while the resources required to process the BINIT# are owned by another logical processor in the same socket may result in a PAL MCA handler hang.

**Implication:** This issue may result in a system hang.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 139. PAL is affecting ITP's ability to halt logical processors

**Problem:** Using ITP to set a breakpoint on one logical processor is not halting all logical processors belonging to that Montecito processor socket.

**Implication:** Logical processors may not halt as expected.

**Workaround:** None at this time

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 140. PAL_MC_ERROR_INJECT not working as expected in the inject_only mode and the inject_and_consume mode

**Problem:** In the inject_only mode or the inject_and_consume mode, PAL_MC_ERROR_INJECT sometimes returns a random value that is not listed in the architectural specification. The query mode is not affected by this issue.

**Implication:** A random value is observed instead of the expected return value.

**Workaround:** If the value returned does not match one of the status values described in the architectural specification, interpret the return status as “Call completed without error.” Negative status values returned (error conditions) are valid as described in the architecture specification.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 141. PAL incorrectly change the value for isr.code

**Problem:** When running under Virtual Machine Monitor software, PAL sometimes incorrectly changes the value for isr.code when a general exception fault occurs.

**Implication:** This issue may result in undefined behavior for the VMM.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 142. Illegal operation faults of the type .lx are incorrectly delivered to the VMM virtualization fault vector

**Problem:** When running under Virtual Machine Monitor software, some instances of illegal operation faults of the type .lx are incorrectly delivered to the VMM virtualization fault vector instead of the expected VMM general exception vector.

**Implication:** This issue may result in undefined behavior for the VMM.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 143. PAL incorrectly routes an illegal operation fault on a reset system mask(rsm) instruction fault

**Problem:** When running under Virtual Machine Management software, PAL incorrectly routes an illegal operation fault on a reset system mask(rsm) instruction fault to the VMM virtualization vector instead of the VMM general exception vector.

**Implication:** This issue may result in undefined behavior for the VMM.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 144. PAL based IA-32 execution does not raise single step trap

**Problem:** PAL based IA-32 execution does not raise single step trap on PSR.ss when EFLAG.tf is enabled.

**Implication:** Debugging of PAL based IA-32 applications may not be possible.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 145. PAL based IA-32 execution does not respond to IA-32 debug traps

**Problem:** PAL based IA-32 execution does not respond to IA-32 debug traps when PSR.db and DBRs are enabled.

**Implication:** Debugging of PAL based IA-32 applications may not be possible.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 146. Intel® Cache Safe Technology "performance restricted" CMCI is issued after 3 ways per set are disabled.

**Problem:** On Montecito processors with 6 MB L3 cache, the Intel® Cache Safe Technology "performance restricted" CMCI only occurs after 3 ways per set are disabled. The CMCI should have occurred after 2 ways have been disabled.

**Implication:** The “performance restricted” CMCI is not occurring after 2 ways are disabled as expected.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 147. Interval Time Counter (ITC) may not be properly initialized

**Problem:** The Interval Time Counter (ITC) may not be properly initialized at reset causing indeterminate operation.

**Implication:** Lockstep mode may not work as expected with this issue.

**Workaround:** This is planned to be fixed in the post production PAL release.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



### 148. POPF instruction may not be intercepted during PAL based IA-32 execution

**Problem:** The POPF instruction may not be intercepted during PAL based IA-32 execution.

**Implication:** The system environment intercept trap may not be taken when CFLG.ii is set to 1.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 149. CMCIs issued noting entry and exit from ETM even when ETM is disabled

**Problem:** When ETM is disabled, the CMCI's will still be issued noting entry and exit from ETM. The processor will not enter ETM, but CMCIs will be issued.

**Implication:** Unexpected ETM related CMCIs are observed.

**Workaround:** Ignore the ETM based CMCI messages if ETM is disabled.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 150. Exclusion of first 3 single bit errors by Intel® Cache Safe Technology may cause system hangs in processors that have their L3 cache size equal to 6MB

**Problem:** The first 3 single bit L3 cache errors will automatically trigger Intel® Cache Safe Technology exclusion. This exclusion is regardless of the type of error (hard error, recurring transient error or standard random error).

**Implication:** This behavior may cause system hangs when using processors that have their L3 cache size equal to 6MB.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 151. Value of the IA-32 interruption code (ISR.code) is incorrectly set

**Problem:** When running under Virtual Machine Monitor software and sending interrupt information to the Virtualization Vector (x6100), the value of the IA-32 interruption code (ISR.code) is incorrectly set to 0x10 when it should be 0.

**Implication:** Depending on the VMM implementation this issue could lead to system hangs in some cases.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 152. Infinite snoop stall during RESET or BINIT

**Problem:** RESET and BINIT flows require the processor caches to be re-initialized and reconfigured from incoherent operation to a coherent operation. If such a transition for a socket is accompanied with data returns intended for that socket within a small window of opportunity, an infinite snoop stall may occur for the next transaction issued on the system bus.

**Implication:** In the case that an infinite snoop stall occurs, the processor may fail to complete the RESET or BINIT flows successfully. Though there exists a non-zero probability of exposure, the issue is highly unlikely to surface in the systems with firmware access times that are longer than 100ns. Systems with low latencies for firmware code fetches that have associated data phases of more than 1 cycle produce sufficient system bus activity for an infinite snoop stall to occur.

**Workaround:** There is no exposure after the caches have been successfully transitioned from incoherent to coherent operation. This is planned to be fixed in the post production PAL release.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 153. Clock misalignment may result in a loss of socket level lockstep

**Problem:** When using parts that have socket level lockstep enabled, there could be a loss of socket level lockstep due to a clock misalignment between the cores.

**Implication:** This issue may result in a loss of socket level lockstep.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 154. Execution of an instruction in the multi-media unit may result in unexpected behavior

**Problem:** When the result of an instruction which uses a multi-media unit is used as an address for a memory instruction, aligns with other internal processor conditions, unexpected L1D CMCI's, L2 Tag BINIT's or other unexpected behavior may result. The conditions necessary to cause these unexpected behaviors are the following:

1. Execution of an instruction in the multi-media unit produces an address that will be used in a memory operation.
2. The timing of the address producer and consumer occurs within a one cycle window.
3. A memory instruction targeting that address hits the L1D cache.
4. Values in the L1D cache and L2D tags contain patterns that create various failure modes.
5. Additional instructions on other clock cycles create electrical conditions that make the memory instruction vulnerable to noise.

**Implication:** Unexpected L1D CMCI's, L2 Tag BINIT's or other unexpected behaviors may be observed. Intel has only observed this behavior in tightly controlled and specifically targeted testing environments and not in an application environment. Intel believes that any compiler or manual code generator targeting performance optimized code will not generate the complete code sequence needed to see this.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 155. Processors may not wake from the LIGHT HALT state upon MCA

**Problem:** After a processor has been placed in the LIGHT HALT state, it will not awaken on a BERR# or a MCA interrupt. The processor will return from the LIGHT HALT state for a BINIT# and other interrupts.

**Implication:** MCA's issued while the processor is in a LIGHT HALT state will be pended and handled when the processor returns from the LIGHT HALT state.

**Workaround:** SAL can rendezvous the processors for global MCA handling.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 156. Logical processor may be lost when a recoverable or a PAL-correctable MCA occurs during PAL_HALT_LIGHT

**Problem:** A logical processor may be lost when a recoverable or a PAL-correctable MCA occurs during PAL_HALT_LIGHT. If the MCA occurs within the 10 instruction bundles immediately preceding the low power halt operation, the PAL_HALT_LIGHT state may not be saved properly when the MCA returns to the interrupted context.

**Implication:** The processor may not return from PAL_HALT_LIGHT.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 157. On Die Termination (ODT) may be unexpectedly enabled

**Problem:** The On Die Termination (ODT) may be unexpectedly enabled when a Montecito Voltage Regulator (MVR) failure occurs or is unexpectedly unplugged.

**Implication:** This may result in undefined behavior on the system bus.



**Workaround:** For testing that includes intentionally removing or powering down an MVR, disable (or tristate) the output of the affected agent on the system bus.

**Status:** For the steppings affected, see the Summary Table of Changes.

### 158. Failure to set the PSR.it bit to its original value

**Problem:** When using PAL_MC_ERROR_INJECT, there may be a failure to set the PSR.it bit to its original value, after an injection event in the register file error consumption mode.

**Implication:** This issue may result in unexpected behavior when using the register file error injection feature in the consumption mode.

**Workaround:** Do not use the register file error consumption mode with PAL_MC_ERROR_INJECT.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 159. The PAL_PSTATE_INFO procedure may write to scratch floating point (FP) registers without saving and restoring the value of PSR.mfl

**Problem:** The PAL_PSTATE_INFO procedure may write to scratch floating point (FP) registers without saving and restoring the value of PSR.mfl around those writes. Therefore, if PSR.mfl was 0 at the entry to PAL_PSTATE_INFO, it may be set to 1 at the exit of that PAL procedure. The PSR.mfl value will most likely always be set to 1 at the exit.

**Implication:** No action necessary.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 160. Performance Monitor Unit (PMU) readings for system interface events may reflect both threads

Problem: When the thread identifier is not driven for system bus transactions, the PMU readings for system bus events will reflect both thread 0 and 1 events.

Implication: PMU readings for system bus events will reflect both thread 0 and 1 events in cases where only thread 0 or thread 1 events should be considered.

Workaround: Include the thread identifier on all system bus transactions to correctly associate thread 0 and thread 1 transactions for PMU system bus events. Ensure that SAL makes the appropriate call to PAL_THREAD_CONTROL to enable this functionality.

Status: See the *Summary Table of Changes* for affected steppings.

### 161. PAL_FREQ_RATIO returns an incorrect value for 1.42 GHz parts

Problem: For 1.42 GHz parts, PAL_FREQ_RATIO returns an incorrect value of 1400:267 instead of 1420:267.

Implication: 1400 MHz is returned for 1420 MHz processors.

Workaround: None

Status: See the *Summary Table of Changes* for affected steppings.

### 162. PAL_MC_CLEAR_LOG called on one logical processor may erase the processor error logs

Problem: For some multiple error scenarios in systems with hyper-threading enabled, PAL_MC_CLEAR_LOG called on one logical processor may erase the processor error log before PAL_MC_ERROR_INFO is called to retrieve the logs.

Implication: This may cause SAL to incorrectly consider an OS-recoverable MCA as fatal.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 163. Unable to specify the Current Frame Load Enable (CFLE) value at the target guest handler

Problem: PAL_VPS_RESUME_HANDLER does not allow the Virtual Machine Monitor (VMM) to specify the CFLE value at the target guest handler.

Implication: PAL_VPS_RESUME_HANLDER disables the CFLE value automatically for the target handling code.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 164. Infinite snoop stalls may be observed

Problem: In extremely rare cases, during the firmware recovery flow, infinite snoop stalls may be observed.

Implication: There may be system hangs observed during the boot flow.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 165. Unexpected behavior when code request completes during PAL authentication

Problem: During boot, if a code request from the system bus or L3 cache completes while any logical processor is authenticating PAL, unexpected behavior may result.

Implication: System hangs during boot or other unexpected behavior may result.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 166. PAL_CACHE_INFO is not available during firmware recovery check

Problem: PAL_CACHE_INFO may not be available during firmware recovery check.

Implication: None.

Workaround: None at this time.

**Status:** See the Summary Table of Changes for affected steppings.

### 167. Potential electrical marginality in the integer register file

Problem: Under certain specific and complex environmental and data conditions a signal race condition can occur that may affect some registers in the integer register file.

Implication: This issue may result in a nested OS fault, an application fault or other unexpected behavior. A nested OS fault may manifest in Unix/Linux as a “Kernel Panic” and in Windows as a “blue screen”.

Workaround: None at this time.

Status: Fixed for processors shipped after September 2006. See the *Summary Table of Changes* for affected steppings.

### 168. PAL_MC_ERROR_INJECT consume mode may not behave as expected

Problem: PAL_MC_ERROR_INJECT consume mode may not cause error consumption and subsequent signaling as expected for level 3 (L3) cache errors.

Implication: PAL_MC_ERROR_INJECT in the consume mode may not work as expected for level 3 (L3) cache errors.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.



### 169. Using PAL_CONTEXT_RESTORE and PAL_CONTEXT_SAVE may result in a system hang during logical processor migration

Problem: Using PAL_CONTEXT_RESTORE and PAL_CONTEXT_SAVE may result in a system hang during logical processor migration.

Implication: None.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 170. PAL_MC_ERROR_INFO may report an invalid index field

Problem: PAL_MC_ERROR_INFO may report an invalid L2D index field for cache_check.

Implication: An invalid L2D index field may be reported.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 171. PAL_BUS_SET_FEATURES bit 52 enables a bus cache line replacement transaction only when a cache line is in the shared state

Problem: PAL_BUS_SET_FEATURES bit 52 enables a bus cache line replacement transaction only when a cache line in the shared state (not the exclusive state) is replaced from the highest level processor cache and is not present in the lower level processor caches.

Implication: PAL_BUS_SET_FEATURES bit 52 may not enable a bus cache line replacement transaction only when a cache line is in the exclusive state.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 172. MOVL instructions taking a general exception fault are decoded as legal virtualized instructions

Problem: When the virtualization environment is enabled, MOVL instructions taking a general exception fault are occasionally decoded as legal virtualized instructions.

Implication: This issue may lead to undefined behavior.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 173. Reserved register field fault checks do not check the present bit to determine if a reserved register field fault should be raised

Problem: When the virtualization environment is enabled, reserved register field fault checks do not check the present bit to determine if a reserved register field fault should be raised for the ITC.i instruction.

Implication: This issue may lead to unexpected behavior.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 174. Calling PAL_CAR_INIT in cacheable mode may cause undefined behavior

Problem: Calling PAL_CAR_INIT in cacheable mode may cause entries in the Cache As RAM (CAR) buffer to be evicted. Specifically, if the CAR buffer size combined with the size of the cacheable code executing outside the buffer, exceed the total size of the L3 cache, the cache lines holding the CAR buffer may be evicted.

Implication: This issue may lead to undefined behavior.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 175. Poison data in the caches has partial or no indication of 2xECC error when written back to memory

Problem: There are different signatures observed in the Itanium 2 3M/6M/9M processors versus the Dual Core Itanium 2 processors.

Problem: Itanium 2 3M/6M/9M processors: During the PAL processing of consumed poison MCA, data from the cache is purged from the cache lines. This issue may cause L2 resident modified poisoned data to not be marked as 2xECC error when written back to memory. Dual Core Itanium 2 processors: L2 and L3 resident modified poison data will only be partially marked as 2xECC error when written back to memory. 2xECC indication is only applied to even 8 byte chunks and the odd 8 byte chunks will not have the 2xECC indication.

Implication: This issue may lead to uncontained poison data.

Workaround: For the Itanium 2 3M/6M/9M processors, there will be a PAL release with the fix. For Dual Core Itanium 2 processors, bit 53 of PAL_PROC_SET_FEATURES (feature set 0) can be set so that the processor signals an MCA when poison data is received by the processor as a PAL corrected error has occurred to facilitate containment.

Status: See the *Summary Table of Changes* for affected steppings.

### 176. Multiple BINIT# assertions due to internal processor timeout (XPN) events

Problem: When a processor XPN timeout event occurs, it will signal multiple BINIT# assertions (if enabled). After the first BINIT, successive unexpected BINITs will occur at fixed intervals until another XPN timeout period elapses. After two XPN timeout periods, the processor will proceed to the PAL MCA handler. This issue is currently masked by E126.

Implication: Multiple BINIT# assertions may be observed due to XPN timer events if/when E126 is fixed.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 177. MSR_L0I_CONFIG corruption during PAL_CACHE_INIT, PAL_CAR_INIT, and reset

Problem: PAL_CACHE_INIT and PAL_CAR_INIT may corrupt the MSR_L01_ICONFIG bit that affects parity error checking in L01.

Implication: May cause loss of parity checking in L01. The system may not raise CMCIs on parity errors.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 178. PAL_HALT_INFO returns an inaccurate value for power savings information

Problem: PAL_HALT_INFO returns an inaccurate value for power savings

Implication: The power savings returned is higher than the actual power savings.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 179. PAL_SET_HW_POLICY uses uninitialized register to initialize thread priority

Problem: PAL_SET_HW_POLICY results in an uninitialized value being used by msr_ebl_thread_1 for the “high” and “exclusive high” priority settings.

Implication: May have a performance effect on systems that call PAL_SET_HW_POLICY to control multithreaded performance settings.

Workaround: None at this time.



Status: See the *Summary Table of Changes* for affected steppings.

### 180. PAL_SET_HW_POLICY may not preserve predicate bit p5

Problem: PAL_SET_HW_POLICY may not preserve predicate bit p5.

Implication: Predicate bit p5 may become corrupted if one thread has set the policy to exclusive, and then a subsequent thread attempts to modify the policy.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 181. PAL_MC_RESUME clears branch registers b6, b7

Problem: PAL_MC_RESUME clears branch registers b6 and b7.

Implication: This could cause undefined behavior.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 182. Snooped L3 tag and/or state ECC error sometimes reports wrong address

Problem: If a snoop that detects an error in the L3 tag or state (any way) is followed closely by a snoop that does not incur an error, the second snoop address is incorrectly logged as the error address.

Implication: 1x or 2x ECC errors may be logged with an incorrect error address.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 183. PAL_PSTATE_INFO returns data not compliant with the SDM

Problem: The two 64 bit blocks of each pstate_buffer entry are in the opposite order of what is specified in SDM revision 2.2 and later.

Implication: PAL_PSTATE_INFO returns data in an unspecified format.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

### 184. Calls to PAL_MC_ERROR_INFO could cause a processor hang

Problem: Saving of registers R7 and R12 in PAL_MC_ERROR_INFO may cause a NaT fault, and because psr.ic is 0 at the time of the fault the interrupt context cannot be resumed.

Implication: Calls to PAL_MC_ERROR_INFO may hang the processor if the NaT bits are set for R7 or R12.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected steppings.

# 7 Specification Changes

**There are no Specification Changes** for this revision of the ***Intel® Itanium® 2 Processor Specification Update***.

# 8 Specification Clarifications

**There are no new Specification Clarifications for this revision of the *Intel® Itanium® 2 Processor Specification Update*.**

## 1. Error logging of deferred IPIs

In the case that an IPI is deferred by the processor and the chipset responds to the deferred IPI with a hard-fail response in the deferred reply transaction, the processor will not log or generate an MCA associated with the hard-fail. Hard-fail response to the deferred IPI can, however, be logged by the chipset.

## 2. Branch prediction across the 40-bit boundary

Chapter 7, of the *Intel® Itanium® 2 Processor Reference Manual for Software Development and Optimization,* May 2004, details *Branch Instructions and Branch Prediction*. The following clarification will be added to the introduction of Chapter 7.

- "A branch prediction across a 40-bit boundary may result in an incorrect target prediction on Itanium 2 processors. Please refer to Table 4-2 for branch prediction latencies in such cases."

## 3. PAL_FREQ_RATIOS in the Intel Itanium Architecture Software Developers Manual, revision 2.2

This is to clarify that when the PAL procedure PAL_FREQ_RATIOS is used for the Dual-Core Intel Itanium-2 processors and the ratio reported is not approximated to the second decimal, the resulting value may be slightly less than the expected integer ratio. For example, when using parts with a maximum frequency of 1.6GHz, the ratio reported is 1600:267. When not approximated to the second decimal place, the actual value of this ratio will be slightly less than 6:1. As a result, if the system bus frequency used for the calculation of the maximum core frequency is less than 267MHz, the calculated core frequency may be slightly less than 1.6GHz.

# 9 Documentation Changes

**There are no new Documentation Changes for this revision of the *Intel® Itanium® 2 Processor Specification Update*.**

## 1. PAL_MC_ERROR_INJECT err_data_buffer description change

In the SDM 2.2, Vol. 2, PAL_MC_ERROR_INJECT procedure, the last parameter, description of 'err_data_buffer' reads:

64-bit physical address of a buffer providing additional parameters for the requested error. The address of this buffer must be 8-byte aligned.

The description will be changed to:

Unsigned 64-bit integer specifying the address of the buffer providing additional parameters for the requested error. The address of this buffer must be 8-byte aligned.

## 2. PAL_MC_ERROR_INJECT procedure err_struct_info - Register File change

In the SDM 2.2, Vol. 2, PAL_MC_ERROR_INJECT procedure, Table 11-95 err_struct_info - Register File

The “Bits” column of “reg_num” will change from (8 bits required):

11:5

to:

12:5

The “Bits” column of “Reserved” (row below reg_num) will change from:

31:12

to:

31:13

# 10 Errata (IA-32 Execution Layer)

**There are no new IA-32 Execution Layer Errata for this revision of the *Intel® Itanium® 2 Processor Specification Update*.**

## 1. Ordering of loads and stores

**Problem:** IA-32 execution layer (EL) reorders IA-32 loads and stores during code optimization. In versions 4 and 5 of IA-32 EL, under some conditions, IA-32 applications executing on IA-32 EL that share memory between processes, or explicitly setting affinity for running logical processors, may not maintain processor ordering of loads and stores.

In version 5 of IA-32 EL, loads and stores of different threads in the same process are guaranteed to follow the processor-ordering rules, except the case that the application explicitly sets the affinity mask of a running thread, and except for floating-point operations (fld and fst instructions) that may expose weak ordering behavior. Integer memory accesses of the same process keep processor-ordering both between themselves and with respect to floating-point memory accesses.

**Implication:** Multiprocessor or multithreaded IA-32 applications that share memory between processes or explicitly set the affinity mask, and in addition depend upon processor ordering, or use fld and fst IA-32 instructions as synchronization semaphores, may not behave as expected. Locks, semaphores, and all other fencing instructions maintain strong ordering and have no exposure to this erratum. Intel has not been able to reproduce incorrect program behavior due to this erratum with commercial software.

**Workaround:** Multiprocessor or multithreaded IA-32 applications should protect access to shared variables with locks, semaphores, or OS synchronization.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

## 2. Segmentation not supported

**Problem:** IA-32 execution layer does not support segmentation, and only limited support for segmentation registers is provided.

**Implication:** IA-32 applications that use segmentation may not operate as expected when executing on IA-32 execution layer. Check with your OS vendor to determine if segmented IA-32 applications are supported.

**Workaround:** IA-32 applications should use the flat 32-bit addressing.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 3. 16-bit application mode not supported

**Problem:** IA-32 execution layer does not support 16-bit application mode. The size address prefix (0x67) is supported only for allowed segment overrides.

**Implication:** IA-32 applications running on IA-32 execution layer that use 16-bit application mode may not behave as expected. IA-32 execution layer does support 16-bit instructions.

**Workaround:** IA-32 applications should use 32-bit application mode.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 4. IA-32 floating-point state

**Problem:** FPUDataPointer, FPUInstructionPointer, and FPULastInstructionOpcode fields of the floating-point (FP) state are not updated by the `FSAVE`, `FNSAVE`, `FXSAVE`, `FSTENV`, and `FNSTENV` instructions.

**Implication:** IA-32 code running on IA-32 execution layer using `FSAVE`, `FNSAVE`, `FXSAVE`, `FSTENV`, or `FNSTENV` instructions cannot retrieve FPUDataPointer, FPUInstructionPointer, and FPULastInstructionOpcode fields from the last non-control FP instruction using these instructions. The last FP state is guaranteed only upon unmasked FP exceptions.

**Workaround:** To get FP state on exceptions, one needs to use the OS-provided context. For example, the user can get the exception record from Windows or use sigcontext on Linux.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 5. Floating-point C1 condition code flag support

**Problem:** IA-32 execution layer does not set the floating-point C1 condition code flag when the last rounding by the instruction was upward. Other C1 behavior is unaffected.

**Implication:** IA-32 code running on IA-32 execution layer that depends upon the C1 condition code flag to identify upward rounding may not behave as expected.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 6. IA-32 floating-point pseudo-denormal, pseudo-NaN, and pseudo-infinity support

**Problem:** IA-32 execution layer will treat pseudo-denormal, pseudo-NaN, and pseudo-infinity values as un-normalized numbers, normalize them, and continue operation rather than raise a denormal exception.

**Implication:** IA-32 code running on IA-32 execution layer using pseudo-denormal, pseudo-NaN, and pseudo-infinity values may not behave as expected. Note that IA-32 processors since the Intel® 387 math coprocessor do not generate pseudo-denormal, pseudo-NaN, and pseudo-infinity values.

**Workaround:** IA-32 applications should avoid using floating-point encodings not supported by the final version of the IEEE Standard 754.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 7. Behavior of quiet and signaling NaNs

These NaN operations have the following behavior:

1. Floating-point operations involving an SNaN operand and a QNaN operand will return a QNaN with the significand of the lesser operand. When moving values using `FLD` followed by `FSTP`, IA-32 execution layer may not convert SNaNs to QNaNs.
2. SSE operations performed on a pair of XMM registers that contain QNaN values may result in the destination changing to the resultant QNaN.

**Implication:** IA-32 code running on IA-32 execution layer that depends upon SNaN or QNaN behavior may not behave as expected.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 8. IA-32 floating-point exceptions

**Problem:** On a FP exception, IA-32 execution layer will set the denormalized operand exception flag when a denormal value has been stored and will set the inexact precision exception flag when an unmasked overflow/underflow fault occurs.

**Implication:** IA-32 code running on IA-32 execution layer depending upon the denormalized or inexact precision flags may not behave as expected.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 9. Partial support for EFLAGS

**Problem:** IA-32 execution layer supports the ID, OF, DF, SF, ZF, AF, PF, CF, and TF EFLAG bits. The IF flag is held to 1. The VIP, VM, and IOPL flags are held to 0. The AC, NT, and RF flags can be written and read by POPF and PUSHF operations, but their semantics are not simulated.

**Implication:** IA-32 code running on IA-32 execution layer depending upon privileged EFLAGS state or the AC, NT and RF flags may not behave as expected.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.



## 10. EFLAGS and floating-point exception flag behavior

**Problem:** EFLAG and FP exception flags may have incorrect behavior when read from an exception handler context, when read from another thread or process, or read by self-modifying code if the flags are not consumed in the original context.

***Note:*** EFLAG and FP exception flags are correct under the use of a debugger.

**Implication:** Multiprocess, multithreaded, or self-modifying IA-32 code running on IA-32 execution layer reading EFLAGS or FP exception flags may not behave as expected if the flags are not consumed in the original context.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 11. RSM and IRET instructions raise incorrect faults

**Problem:** On IA-32 execution layer, `RSM` calls raise a general protection fault, and `IRET` calls raise an illegal operation fault.

**Implication:** These are not expected to occur in user mode.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 12. Cross-modifying code

**Problem:** IA-32 execution layer may not maintain execution consistency of multiprocess cross-modifying IA-32 code if a process has opened the instruction page with read-only permission.

**Implication:** Multiprocess cross-modifying IA-32 applications may not behave as expected, if a process has opened the instruction page with read-only permission.

**Workaround:** Multiprocess cross-modifying IA-32 applications should open modified instruction pages with read/write access.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 13. Atomicity of lock-prefixed instructions making unaligned memory references

**Problem:** On IA-32 execution layer, an IA-32 lock-prefixed instruction making an unaligned memory reference is performed atomically only with respect to other lock-prefixed instructions making unaligned memory accesses in the same process.

**Implication:** If an unaligned memory access is made to the same physical address by a lock-prefixed instruction and another process, an instruction without a lock prefix, or an aligned lock-prefixed instruction, atomicity is not guaranteed, and the code may not behave as expected.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

## 14. Atomicity of lock-prefixed instructions making uncacheable memory references

**Problem:** On IA-32 execution layer, an IA-32 lock-prefixed instruction making an uncacheable memory reference is performed atomically only with respect to other lock-prefixed instructions making uncacheable memory accesses in the same process.

**Implication:** If an uncacheable memory access is made to the same physical address by a lock-prefixed instruction and another process, an instruction without a lock prefix, or an uncached lock-prefixed instruction, atomicity is not guaranteed and the code may not behave as expected.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

### 15. Noninterruptability of 32-bit unaligned and 16-byte stores

**Problem:** On IA-32 execution layer, if a thread is suspended during a 32-bit unaligned or a 16-byte IA-32 store to cached memory, another thread may observe partially updated memory until the OS can service the thread suspension.

**Implication:** When a process performs 32-bit unaligned or 16-byte stores, partial memory updates may be observed by other threads until the OS can service the thread suspension, resulting in unexpected behavior.

**Workaround:** None at this time.

**Status:** For the versions affected, see the *Summary Table of Changes*.

### 16. IA-32 execution layer install and uninstall failures

**Problem:** On some Itanium 2-based platforms, incorrect reports may be seen while installing or uninstalling IA-32 execution layer.

**Implication:** During installation, the IA-32 execution layer installer “IA-32ExecutionLayerSetup.exe” may incorrectly report that a previous version has been installed and ask the user to remove the previous installation.

After an uninstall and subsequent reboot, the system may incorrectly ask users to reinstall IA-32 execution layer.

**Workaround:** Users should download the latest IA-32 execution layer installer “IA-32ExecutionLayerSetup_1.exe”(revision 1 or greater) from the Microsoft* download center.

**Status:** For the versions affected, see the *Summary Table of Changes*.

### 17. Self-modifying code on unaligned memory may result in an access violation

**Problem:** If an IA-32 application contains a basic code block that;

- Is doing self-modifying code,
- That modifies the very first instruction of the basic code block, and
- This basic block accesses an unaligned memory address.

Then the application may crash with an access violation (general protection fault).

**Implication:** Applications that use a self-modifying basic block on unaligned memory addresses may fail and result in a general protection fault.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 18. Large data file accesses may return incorrect data

**Problem:** When a Linux-based IA-32 application running on IA-32 execution layer tries to access a file-offset beyond 4 GB, the wrong data may be accessed.

**Implication:** The use of incorrect data may result in unpredictable system behavior.

**Workaround:** None at this time. This issue is fixed in version 5.3.81.31.21 and above.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### 19. IA-32 EL applications will not run on kernels with page sizes greater than 16k

**Problem:** When a Linux kernel is built using a page size greater than 16 k, IA-32 applications utilizing the IA-32 execution layer will not load or run.

**Implication:** IA-32 processes that are loaded will immediately crash.

**Workaround:** None at this time. This issue is fixed in version 5.3.88.34.22 and above.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



## 20. IA-32 EL may incorrectly optimize frequently executed code with interleaved integer and floating-point flag operations that include producer/consumer code sequences

**Problem:** The IA-32 EL may optimize frequently executed instructions with a faster code sequence. A special case exists when the following conditions are met:

- The IA-32 application code includes floating-point and integer code sequences that use flags in a producer and consumer programming model.
- The iterations of this code are large enough to benefit from IA-32 EL performance optimization.

In the case where the integer and floating-point code is intermixed it is possible under certain conditions for the code optimization to incorrectly translate the flags used by the consuming code.

**Implication:** IA-32 application code with intermixed integer and floating-point flag producer/consumer sequences may fail with unpredictable results if this code is optimized by IA-32 EL. This erratum affects versions 5.3.5336 to 5.3.5337 of IA32Exec.bin. Version 5.3.5338 of IA32Exec.bin contains a fix for this erratum.

**Workaround:** If possible avoid mixing the integer and floating-point code sequences used in a producer/consumer programming model.

**Status:** For the steppings affected, see the *Summary Table of Changes.* Version 5.3.5338 of IA32Exec.bin contains a fix for this erratum.

## 21. IA-32 code running with the IA-32 EL may see an SSE Exception being ignored after the FPREM1 instruction is executed

**Problem:** In the case where the following conditions are met:

- IA-32 code is being executed through IA-32 EL.
- The IA-32 code clears one or more of the SSE exception mask bits in the MXCSR register.
- The corresponding FCW register bit or bits are set.

When the IA-32 code is executed and calls the FPREM1 instruction the bits in the emulated mask in the MXCSR register may not be cleared until after FPREM1 has completed its execution.

**Implication:** An IA-32 procedure that turns off a bit in the MXCSR register SSE exception mask but fails to do the same for the corresponding FCW bit, may find that if an exception occurs during the execution of the FPREM1 instruction the exception could be ignored. This erratum affects IA-32 EL version 4 and 5, fixed in version 5.3.5338 of IA32Exec.bin.

**Workaround:** The setting or clearing of the exception mask bits should match in both the FCW and MXCSR registers or avoid using the FPREM1 instruction.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

## 22. An IA-32 EL optimized code procedure with interleaved MMX™ and SSE code may experience an application hang

**Problem:** The IA-32 EL may optimize frequently executed instructions with a faster code sequence. If the code loop contains interleaved MMX™ and SSE instructions under complex and rare conditions the application may hang.

**Implication:** An IA-32 application that mixes SSE and MMX code in a frequently executed code procedure may experience an application hang. Affects IA-32 EL version 5, fixed in version 5.3.5338 of IA32Exec.bin.

**Workaround:** Avoid mixing MMX and SSE instructions in the same code loop.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 23. An IA-32 Linux* application may receive an unexpected memory access violation

**Problem:** If an IA-32 Linux application is running through the IA-32 EL and an interrupt is taken during code execution, due to the way that stack space is allocated the application may receive an unexpected error.

**Implication:** If the current memory stack pointer is close to the end of the allocated stack space an interrupt received during the execution of an IA-32 Linux application may generate an unexpected memory access violation and terminate execution of the program. This erratum affects libia32x.so version 5.3.74.27.29 to 5.3.98.37.22

**Workaround:** None available at this time.

**Status:** For the steppings affected, see the *Summary Table of Changes.*

### 24. Wrong NEG EFlags cases

**Problem:** Some combinations of NEG instruction, flag consumer, and register values can cause an incorrect flag to be emulated. If an application executes the NEGW reg instruction (16-bit operand size), and the value of reg is 0x8000, producing the SF, and this flag is consumed afterwards, then the emulated SF value can be incorrect.

If the application executes the NEGB reg instruction, when reg = *H (AH/BH/CH/DH) and the corresponding *X value falls under one of the following two cases:

- *H=0 and *L!=0 [e.g. BX=0x0003]
- *H!=0 and *L=0; [e.g. DX=0x1000]

and this NEGB *H instruction produces the ZF or CF flags, which are consumed afterwards by SETcc/Jcc/CMOVcc/FCMOVcc where cc = ae/b/e/ne, then the emulated CF / ZF value can be wrong.

**Implication:** An unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 25. Lock XADD atomicity

**Problem:** Lock XADD executed as non-atomic during IA-32 EL interpreter phase (only at the beginning of the execution). If an application uses lock XADD to perform inter-thread synchronizations, the atomicity of the operation cannot be guaranteed during the first tens execution of this lock XADD instruction.

**Implication:** Hang or other unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 26. Lock <***> + MOV weak order

**Problem:** A hyper-threaded application, running on top of an MP LINUX* platform, demonstrating genuine thread-parallelisms and executes a LOCK XXX instruction as specified below, immediately followed by a load from memory (for example, MOV reg, [mem]), can view these two accesses as weakly ordered:

** =

ADC,ADD,SBB,SUB,INC,DEC,NOT,OR,XOR,AND,NEG,BTC/R/S,XADD,XCHG,CMPXCH

The bug is exposed only during the first several thousands of execution of this code.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.



## 27. SSE with behavior change

**Problem:** Possible wrong exception / suspension state in SSE code after behavior change. Consider 3 code portions A,B,C, all containing SSE instructions. Assume all 3 run several thousands of times, and later on the frequent internal paths between them change. e.g., A --> B is dominant at first 5K B entrances, but afterwards, it is C --> B. In some cases, IA-32 EL may choose to modify the translation of B, this modified translated B wrongly responds to exception or suspension – it can reconstruct an incorrect IA-32 context.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

## 28. Thread not suspended

**Problem:** A thread-suspension can return as successful although the thread is running (Linux only). When a hyper-threaded application, running on top of an MP LINUX platform, demonstrating genuine thread-parallelisms and calls for thread suspension (either by a kernel API or by using a threading library call), there is an extremely low chance that the request will return a success indication but the thread will still be running. It happens if a thread T1 is suspending a thread T2, and the following race condition occurs: T2 is returning from a system call, and executing a specific gate unlock inside BTGeneric (a very specific point inside BTGeneric), and exactly at the same time thread T1 is trying to take this lock, there is a small chance that T2 will “believe” it succeeded to block T1 from entering BTGeneric (in order to continue emulation), while in fact T1 already passed this point and continues execution. As a result, T2 may “believe” it succeeded to suspend T1 in the 32-bit sense – while in fact T1 executes its IA-32 instructions. So suspension success indication is returned, while the suspension actually did not take place.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

## 29. Extended-double to double precision

**Problem:** An extended double-precision fmul followed by double precision fst may result in a slight precision deviation. If an application performs a floating-point multiplication with extended precision that is followed by a store into double-precision element in the memory, in some rare cases the result in memory can slightly deviate from the IA-32 compatible result.

The bug is exposed only after the first several thousands of execution of this specific code. Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

## 30. CMPXCHG EAX, reg

**Problem:** Wrong CMPXCHG EAX/AL/AH, REG result during IA-32 EL interpreter phase. If an application executes the CMPXCHG EAX/AL/AH,REG instruction, the eax/al/ah value may incorrectly be kept unchanged in the first tens execution of this instruction.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

## 31. SSE with early loop exit

**Problem:** Possible incorrect XMM register content when exiting from an early exit of a loop. If an IA-32 application contains a code sequence which:

1. Forms a loop.
2. Contains SSE (1/2/3) instructions.
3. Has at least one early exit (a conditional jump leaving in the middle of the loop); and this early exit is mostly untaken at the first several thousands of iterations of the loop body.
4. An XMM register is written after the early exit, and this register is not read or written between the loop-body-entrance and this early exit.
5. After several thousands of iterations of the loop body, there is an execution instance when the early exit is taken immediately after at least one full iteration is executed.

Then, it is possible that this XMM register will contain the wrong value after the exit.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

## 32. Exception/suspension in fnstsw-sahf-jcc

**Problem:** Wrong value of EAX can be reconstructed after exception/suspension occurs inside an fnstsw-sahf-jcc sequence. If an exception or suspension occurs inside an fnstsw-sahf-jcc sequence, in a hot block, a wrong value of EAX may be reconstructed:

- In most cases IA-32 EL chooses to restore the state before the fnstsw, and then the operation is OK.
- If the restored state is between the SAHF and the Jcc, only the EAX is wrong, but the Jcc is OK. In most cases, it will not cause any visible effect.
- If the restored state is between the FNSTSW and the SAHF, both the EAX and the Jcc are incorrect.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

## 33. Load-misalign-reload

**Problem:** Wrong content loaded from memory in case it is misaligned and placed between two identical loads overlapping with the other access. If an application executes the following sequence Load; Misaligned Lock RMW; Reload; or Load; Misaligned Load; Reload, and the following conditions hold:

1. The load and the reload can be statically identified as accessing exactly the same address
2. The intermediate access is misaligned
3. The intermediate overlaps the loads' address
4. Both loads are integer or both FP
5. Then, the value read at the reload may be incorrect.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

## 34. Incorrect register values in multi-block prefetch

**Problem:** Register corruption in some cases of dynamic data prefetch in multi-chain loops.

When:



1. An IA-32 IP is translated in more than one chain (hot block) on same multi-chain loop
2. There is a dynamic data prefetch generated for that IA-32 IP
3. The number of IA-32 IPs for which a dynamic data prefetch is generated for exceeds 5, then IPF integer register#80 will be overwritten.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 35. Suspension while SMC observed

**Problem:** Wrong IA-32 state can be restored when a thread that inspected modified code is resumed from suspension. Suppose a thread T1 that is executing a hot loop on writable-page code, is suspended by another thread T2, and just upon resume it detects that Self Modifying Code (SMC) has occurred in the loop body. In such a case, a wrong state may be reconstructed.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 36. LINUX internal synchronization

**Problem:** LINUX internal synchronization object is not released properly. In the code that releases an internal sync object, the order of memory barriers is not correct.

**Implication:** Unpredictable failure.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 37. Page crosser lock w/ permission change

**Problem:** If one thread performs a page-crosser locked instruction while another thread is changing the write permission of one of the pages, the instruction may be viewed as non-atomic and an extra access violation may occur. A hyper-threaded application in which one thread performs an unaligned lock access which also crosses pages, while one of these pages is a subject to a page-permission change in another thread, and the application could have recovered from the access violation by an exception handler, the memory may contain a partial write and 2 access violation events may be observed instead of a single one.

**Implication:** Unpredictable failure.

**Workaround:** None at this time.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 38. Socketcall send/receive message may fail

**Problem:** If an IA-32 Linux application uses an OS socket system (socketcall) communication message and the size of the message buffer is greater than 1 (msg_iovlen > 1), then the socket message (sendmsg or recvmsg) may fail.

**Implication:** In most failed cases the system call will indicate a failure but in some extreme cases the result of a failure may be unpredictable.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 39. Interrupted long Linux system call that receives an interruption-indication may unexpectedly modify an application buffer

**Problem:** An IA-32 Linux application may unexpectedly find its buffer modified under the following conditions:

- The application issues a long Linux system call (a system call that can return due to an asynchronous signal) that includes (pointers to) an initialized input buffer.
- This same application thread receives a system interrupt concurrently or soon after the long system call is made, but before the system call reaches the kernel itself.
- The same thread receives a second system interrupt while the thread is inside of the kernel and this second interrupt makes the kernel return an 'interrupted' indication to the thread without executing the system call.

Under these conditions the contents of the application's input buffer may be partially or completely over written.

**Implication:** The result of this erratum is unpredictable and is dependent upon the use of the application and the application buffer.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 40. ZF flag may be mishandled when using a CMPXCHG8b in an If-Then-Else code structure

**Problem:** The Zero Flag (ZF) may be unexpectedly consumed and cleared under the following conditions:

- An application sets the ZF flag and enters an If-Then-Else code structure.
- One side of the If-Then-Else code block contains a CMPXCHG8b instruction, and
- The other side of the code structure does not use or change the ZF flag.
- The ZF flag is consumed after the If-Then-Else code structure is exited.
- The application code block is a frequently executed code sequence and is optimized by IA-32 EL.

Under these conditions, the ZF flag may incorrectly be read as cleared after the application has passed through the side of the If-Then-Else code structure that does not contain the CMPXCHG8b instruction.

**Implication:** This issue has only been observed in a synthetic test environment. The results of this erratum are unpredictable and dependent upon the use of the affected application.

**Workaround:** None at this time. This erratum is fixed in version 5.3.108 of the libia32x.so system library file.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 41. Performing SSE divide of a denormal value by zero, while the DAZ bit is set, will result in a zero-divide exception instead of invalid-operation exception

**Problem:** If an IA-32 application, running on top of IA-32 EL, performs an SSE divide (DIVPD, DIVSD, DIVPS or DIVSS) where the denominator is zero and the numerator is a Denormal value, a zero-divide exception will be raised instead of an invalid-operation exception.

**Implication:** If an IA-32 application turns DAZ on, and relies on specific exception type (invalid-operation exception, which is raised when dividing zero by zero) to be raised, it can fail. If the programmer suspects that the denominator may be zero, they should protect both cases.

**Workaround:** A program that intends to handle zero division and uses DAZ, should check both for invalid operation and zero-divide.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 42. Asynchronous suspend and resume calls to a thread may result in undefined behavior

**Problem:** If an IA-32 Windows*-based application, running on top of IA-32 EL contains at least three running threads T1, T2, T3; and T1 is trying to suspend T3, while at the same time T2 tries to resume it (although T3 was not suspended), the results are undefined.

**Implication:** If an IA-32 application performs suspend-thread and resume-thread in an asynchronous manner, including resuming running thread while another thread attempts to suspend it, may fail in an unpredictable way. In most cases, this will result in a process crash.

**Workaround:** The program should serialize the suspend-resume request for any of its threads or avoid resume-thread calls to threads that are not suspended.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 43. Files under /proc/<pid> may contain incorrect data for emulated processes

**Problem:** When examining /proc/pid, where 'pid' is an IA-32 process emulated by IA-32 EL, some of the data may be incorrect.

When examined by the current process, the following fields will hold incorrect data:

- 'exe'
- 'statm'
- 'status' / memory and signal related fields
- 'maps'

When examined by another process, most fields will show incorrect data.

**Implication:** Applications that rely on examining data of other processes (or specific fields for the same process) through /proc interface may fail.

**Workaround:** None.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 44. Select pending signals and SIG_IGN dispositions are not inherited cross-execve

**Problem:** Pending signals of the following types:

- SIGSEGV
- SIGBUS
- SIGFPE
- SIGILL

will not be inherited to new context after executing execve. The same is true for their disposition.

**Implication:** Applications that rely on previous context for having the pending signals ready for them, or their disposition will not get these signals and/or their disposition.

**Workaround:** Programs should not rely on inheritance, across execve call, of pending signals that are also HW events, and neither on their disposition.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

### 45. Floating-point content reuse

**Problem:** Problem: If an IA-32 application, running on top of IA-32 Execution Layer, performs a sequence of

- FP/SSE Loads of a value, X, from an address on the stack, A, calculated based on register R1
- Store to B calculated based on another register R2, B overlapping with A
  - — The calculation of B involves big back and forth offset from A, such as R2 <- R1 + K, B = R2 - K, |K| > 0x4000
- Second FP/SSE Load from A (not based on R2)

Then, under some internal IA-32 EL conditions, IA-32 EL may reuse X as the result of the 2nd load.

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 46. FXSAVE with extensive SSE and floating-point usage may use incorrect values from the XMM registers

**Problem:** If an IA-32 application performs a piece of code that:

- contains an FXSAVE instruction,
- contains SSE instructions,
- contains FP instructions,
- a rare internal condition of FP register pressure occurs in the block,
- and this code is in a hot block

Then, one or more of the XMM values can be wrongly saved in memory by the FXSAVE instruction.

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 47. Interruption of a loop with SSE may incorrectly restore XMM registers

**Problem:** If an IA-32 application

- performs SSE code in a loop,
- and this code is executed for several thousands of times,
- and it is interrupted by suspension or exception inside the loop

Then, on some internal conditions, the XMM register values (or part of them) may be restored with wrong values.

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 48. Unmasked numeric FP exception in FXTRCT may view wrong FP values

**Problem:** If an IA-32 application

- operates an a piece of code rich in FP operations,
- one of them is FXTRACT,
- and the same piece of code runs for several thousands of times, when at least one of #I, #Z or #D exceptions, is unmasked,
- and then an unmasked numeric exception (#I/#Z/#D) occurs in this FXTRACT operation,

- and an exception handler consumes the FP register values

Then, on some rare internal conditions, an FP register value may be wrong

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 49. On rare conditions, FP exceptions shortly after an FCLEX/FNCLEX may view wrong status bits

**Problem:** If an IA-32 application

- operates an FCLEX or FNCLEX operation,
- and the same piece of code runs for several thousands of times,
- and then an exception occurs in this piece of code,
- and an exception handler consumes the FP status exception flags

Then, on some very rare internal conditions, an exception flag may be set although it should have been cleared by the FCLEX

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 50. An unmasked inexact SSE exception on some instructions may not be restored correctly

**Problem:** If an IA-32 application

- performs one of the following instructions that contains an MMX register as destination and SSE value (in an XMM register or in memory) as source, or vice versa: CVTPS2PI, CVTTPS2PI, CVTPI2PS, CVTPI2PD, CVTPD2PI, CVTTPD2PI, MOVDQ2Q, MOVQ2DQ,
- and this code is executed for several thousands of times,
- while the #I exception control-bit is unmasked,
- and such a #I exception occurs in this instruction,

Then, on some internal conditions, the exception context may be restored with wrong values.

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 51. SSE exceptions in a hot block may incorrectly set flags

**Problem:** If an IA-32 application, running on top of IA-32 Execution Layer, performs an SSE instruction that causes a HW exception, after several thousands times of execution of this block, and under some very rare internal conditions, then

- The C1 flag can be wrongly set to 0
- In case the instruction is *LDMXCSR*, the status bits can also get wrong values

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 52. Multiple exceptions between two code blocks may lead to an incorrect context

**Problem:** If an IA-32 application encounters the following sequence:

- An exception occurs in a “young” piece of code, that is, executed for only a few tens of times
- The application recovers from this exception (handles it) and continues execution
- A short time after, another exception occurs, this time on an “old” piece of code

Then, under some internal conditions, the context of the second exception may be recovered incorrectly.

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 53. Numeric SSE exceptions could be ignored initially after being unmasked

**Problem:** If an IA-32 application encounters the following sequence:

- At some point, all SSE and FP numeric exceptions are masked,
- a block that
  - — Contains an SSE instruction S that may raise a numeric exception
  - — Contains no FP instruction

Runs for several thousands of times under this “mask SSE/FP” condition,

- afterwards, an MXCSR control bit, which is relevant to S, is unmasked (S can raise a numeric exception now),
- and indeed this exception occurs

Then, under some very rare additional internal conditions, the exception is not delivered and the application behaves as if the exception never happened.

**Implication:** Usage of incorrect data may result in unpredictable system behavior.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 54. Application writing to a guarded page on Windows may fail on access violation

**Problem:** If an IA-32 application protects a page by a PAGE_GUARD attribute, then tries to execute it under a structured exception handling closure, and examines the exception code in the exception filter, may wrongly observe access-violation code instead of guard-page-execution.

**Implication:** Applications that rely on the correct exception code when trying to execute a guarded page, may fail with unexpected access violation.

**Workaround:** Do not use guard-page, or if you do, treat AV as a guard-page violation.

**Status:** Fixed

### 55. Job Memory Limit on Windows

**Problem:** If an IA-32 application is running with job memory limit, which is low enough so that IA-32 EL cannot find the room for translations, it may slow down by many orders of magnitude. The application seems to be in a complete hang.

**Implication:** Applications that run in such conditions hang.

**Workaround:** Do not use too severe job memory limit.

**Status:** Fixed

### 56. Reloading a modified DLL may fail

**Problem:** If an IA-32 application

- Loads a DLL
- Runs its code (enough time to heat some blocks)
- Unloads it
- Modifies the DLL's disk image (in the same directory)
- Reloads the modified DLL from the original disk place

May behave in an unpredictable manner.

**Implication:** Applications that performs the sequence above, will most probably abort or hang, but can also behave in an unpredictable manner

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 57. Linux* core file generation

**Problem:** If an executable, non-readable application crashes, and the environment is set for core-file generation, a core file is generated and can be accessed by the user.

**Implication:** Parts of the binary may be exposed.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 58. Linux* EXECVE fails to launch NR file

**Problem:** If an IA-32 application executes EXECVE Linux system call for a non-readable application, the EXECVE may return with a failure indication and the process will not be launched.

**Implication:** The second process will not start.

**Workaround:** Upgrade to the latest version. Fix is in version 5, update 1 and later.

**Status:** Fixed

### 59. ptrace returns wrong system-call id

**Problem:** For IA-32 applications running on a Linux*-based operating system, if a debugger queries an IA-32 application being inside a system call, which was the EAX value at the call, it may get -1 instead of the correct system call ID.

**Implication:** The second process will not start.

**Workaround:** Upgrade to the latest version.

**Status:** Fixed

### 60. READV/WRITEV overflow

**Problem:** If an IA-32 application performs a READV or WRITEV Linux* system call, and the sum of all lengths specified in the IO vector input is greater than 2G, the application may abort instead of return with a failure indication.

**Implication:** The application will abort.

**Workaround:** Do not call READV/WRITEV with such a huge vector lengths.

**Status:** Fixed

### 61. More precise FP calculation result

Problem: A FP instruction which operates on single precision number while FPCW.PC is not single precision mode may produce a more precise result that may not be IA-32 bit compatible.

Implication: The application may get an incorrect or undefined result.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 62. Wrong exception flags in interrupted context

Problem: If an IA-32 thread is interrupted while executing X87 FP or SIMD FP instructions, exception flags of FPSW or MXCSR in interrupted context may be incorrect.

Implication: Applications which depend on FPSW or MXCSR exception flags in interrupted context may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 63. Wrong CF/AF in interrupted context for LOCK SBB

Problem: If an IA-32 thread is interrupted while executing SBB instruction with LOCK prefix, CF/AF of Eflags in interrupted context may be incorrect.

Implication: Applications which depend on the CF/AF Flags in interrupted context may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 64. Wrong ZF/PF/SF in interrupted context for AAM

Problem: If an IA-32 thread is interrupted while executing the AAM instruction, ZF/PF/SF of Eflags in the interrupted context may be incorrect.

Implication: Applications which depend on ZF/PF/SF in an interrupted context may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 65. Unaligned RMW instruction interruption handling

Problem: Interruption on RMW (Read-Modify-Write) instructions which access unaligned memory that crosses writable and non-writable pages may cause an unexpected access violation.

Implication: Applications may abort unexpectedly.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 66. Unexpected access violation on PUSH/POP

Problem: XADD/XCHG instruction with two register operand and source operand is ESP may cause unexpected access violation.

Implication: Applications may abort unexpectedly.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 67. Wrong interrupted EIP on instructions consuming PF

Problem: If an IA-32 thread is interrupted while executing instructions that consume PF, the EIP in the interrupted context may be wrong.

Implication: Applications that depend on EIP in an interrupted context may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.



### 68. Interruption in unaligned CMPXCHG

Problem: If CMPXCHG mem, reg instruction, where mem is not naturally aligned, is interrupted, the memory content may be wrong.

Implication: Applications may get undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 69. Wrong exception masks in MXCSR

Problem: An IA-32 thread with unmasked exception bit in both FPCW and MXCSR may get wrong MXCSR in cases where there is a FP stack fault exception.

Implication: Applications that unmask some SIMD exceptions may not get expected exceptions, and the application may get undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 70. Wrong flags in interrupted context

Problem: If an IA-32 thread with unmasked exception bits in FPCW or MXCSR is interrupted when executing floating point comparison instructions, the corresponding flags in interrupted context may be incorrect.

Implication: Applications that use these flags in the interrupted context may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 71. Wrong TOP in interrupted context for SQRTSS

Problem: If an IA-32 thread is interrupted while executing SQRTSS, TOP of FPSW in the interrupted context may be incorrect.

Implication: Applications that use FPSW.TOP in the interrupted context may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 72. Applications unexpectedly abort

Problem: Interruption on instructions that have memory source operand and modify PF may cause an unexpected access violation.

Implication: Applications may abort unexpectedly.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 73. Lock instruction with unaligned memory reference

Problem: If there are two code sections which contain lock instructions with unaligned memory references and a jump from the same predecessor, the lock instruction may produce incorrect results.

Implication: Applications may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 74. Second 4/8/16-byte unaligned load

Problem: If the same 4/8/16-byte unaligned load is executed in different code sections, and the first code section is branched from another code section with Flag producer – Jcc sequence, the second 4/8/16-byte unaligned load may get incorrect results.

Implication: Applications may get undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 75. Wrong FP registers value in interrupted context

Problem: If an IA-32 thread is interrupted while executing FP instructions followed by some specific instructions, the FP registers value in the interrupted context may be incorrect.

Implication: Applications which depend on FP register value in an interrupted context may fail with undefined results.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 76. LOCK NOTW [odd address] negates 4B

Problem: If LOCK NOTW refers to a misaligned (odd) address, the operation negates 32 bits instead of 16. If a LOCK BTSL refers to a misaligned address that is 2-bytes aligned, the process may crash.

Implication: If an IA-32 application uses LOCK NOTW for an odd memory address the results are undefined. If it uses LOCK BTSL for address=4x+2, the process may terminate unexpectedly.

Workaround: Align all memory areas that are accessed by LOCK operations; particularly LOCK NOTW / BTSL.

Status: See the *Summary Table of Changes* for affected versions.

### 77. LOCK RMW suspension atomicity break

Problem: If an IA-32 thread is suspended while executing LOCK Read-Modify-Write memory operation, an inconsistent state may be observed.

Implication: Suspend-resume of a thread while performing a LOCK RMW operation may behave non-atomically. If the application relies on the atomicity of the memory operation, the results are undefined.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 78. Flags on CMPXCHGW

Problem: When CMPXCHGW instruction is executed, and the values of AX and the memory operand has opposite signs, and the CMPXCHGW generated-flags are consumed by Jcc or CMOVcc with signed-inequality semantics (LT/LE/GT/GE), a wrong decision can be taken by the consumer.

Implication: If an application executes CMPXCHGW with the conditions above, the results are undefined.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 79. Wrong ZF on cmpxchg8b

Problem: When CMPXCHG8B / ARPL / LSL / LAR / VER instruction turns the ZF off, after the previous flags-producer generated zero-result with overflow, the ZF may erroneously appear as set to the next consumer; when LSL or LAR instruction updates its destination, but the new value happens to coincide with the old one, the ZF flag is erroneously cleared.



Implication: If an application executes one of the scenarios above, and relies on the correct value of ZF, the results are undefined.

Workaround: Do not use LSL/LAR, or if you do, do not rely on ZF to indicate a write that is not changing the value.

Status: See the *Summary Table of Changes* for affected versions.

### 80. Flags at interrupt after then/else

Problem: Exception or suspension right at the beginning of the “else” part of an “if” construct, or right after it, can view wrong values of flags. The same problem occurs if the exception or suspension comes at the end of CMP-SBB/ADC-MOV [mem].

Implication: For applications that fall under this sequence of conditions, the results are undefined.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 81. DIVPS [m128] interruption crash

Problem: If an IA-32 thread is suspended at the beginning of executing DIVPS with memory source, or if the memory access faults, the application may terminate unexpectedly.

Implication: Suspend-resume of a thread while performing a DIVPS [m128], or allowing the memory access to fault and handle the exception, may result in abnormal termination of the process (abort).

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 82. Crash on optimization sequence

Problem: If 2 consecutive very short code sections are running several thousands of times, on some extremely rare internal conditions the process may terminate unexpectedly.

Implication: Applications encountering these conditions may abort.

Workaround: None at this time.

Status: See the *Summary Table of Changes* for affected versions.

### 83. Ignored Self Modifying BTX

Problem: If an application modifies its own code by using BTS/BTR/BTC as the modifying instruction, and the second operand (bit offset) is greater then the operand size, the modified code may remain unobserved (continue executing the old version).

Implication: If an application relies self modifying code using BTX, the results are undefined.

Workaround: Avoid using BTS/BTR/BTC for self modifying code operation.

Status: See the *Summary Table of Changes* for affected versions.

### 84. Lost signal in spin-loop

Problem: If an application spin-loop waits for a signal to exit, and the signal arrives just after the block is translated, and at the same time another thread reaches the same point, the signal may be lost and the spin loop is executed indefinitely.

Implication: Applications that use spin-loop to wait on a signal may not operate as expected.

Workaround: Time-out spin-loops that are waiting for signals.

Status: Linux only. See the *Summary Table of Changes* for affected versions.

### 85. Debugger aborts on “fail to attach”

Problem: On some rare conditions, when a debugger process tries ptrace-attach to a debugged process, the attach may return a failure, and the debugger terminates unexpectedly.

Implication: The debugger application may terminate unexpectedly.

Workaround: Retry the debugger (The failure is timing-dependent and it is very likely to succeed next time).

Status: Linux only. See the *Summary Table of Changes* for affected versions.

# 11 IA-32 Execution Layer Specification Clarifications

**There are no new IA-32 Execution Layer Specification Clarifications** for this revision of the *Intel® Itanium® 2 Processor Specification Update.*

## 1. Aliasing of MMX registers to FP registers

If a value is written to the FP register, and an MMX™ operation is performed to the corresponding MMX register, the exponent portion of the corresponding FP register may not be written to 1's if the register's significand is unchanged by the MMX instruction.

As described in the *IA-32 Intel® Architecture Software Developer's Manual*, the `EMMS` instruction, which empties the MMX state by setting the tags in the x87 FPU tag word to 11B, must be executed at the end of an MMX routine before calling other routines that can execute FP instructions.

## 2. Floating-point and SSE precision

Floating-point and SSE instructions like `RCPPS`, `RCPSS`, `RSQRTPS`, and `RSQRTSS` may provide slightly more precise results than Itanium 2 processors or IA-32 Intel processors since IA-32 execution layer may merge separate `FADD` and `FMUL` instructions into a single `FMA` instruction or replace two roundings by one rounding. In particular, subtraction of two infinite or 'almost infinite' numbers that originated from a multiplication operation may result in a significant difference than on a native IA-32 processor due to the different rounding effects.

## 3. CPUID values represent the IA-32 execution layer processor model

CPUID return values accurately represent the IA-32 execution layer processor model, but may not represent the physical processor in the system. The vendor and family information are correct for IA-32 execution layer, but cache, translation lookaside buffer (TLB), and other processor-specific information is not supported. The CPUID values returned by IA-32 execution layer will be documented in the *Intel® Processor Identification and the CPUID Instruction Application Note* (AP-485).

## 4. IA-32 execution layer resides in the application virtual address space

IA-32 execution layer components, memory for translated code blocks, and IA-32 execution layer data structures, all reside in the application virtual address space. Memory requests may be denied if insufficient memory is available. Non-relocatable DLLs may fail to load if that memory is already occupied.

## 5. Signal delivery may be postponed during code translation or garbage collection

During code translation or garbage collection, signal delivery may be postponed. There is no maximum time-limit, but the delivery is guaranteed to happen eventually.

## 6. Aborting threads could cause other process threads to hang

An application running on IA-32 execution layer may use internal IA-32 execution layer critical objects. Aborting a thread that holds an IA-32 execution layer critical object could cause the other threads in the process to hang.

### 7. Core dump files cannot be produced correctly when an IA-32 process is aborted

When an IA-32 process is aborted, a core dump file can often be used for debugging purposes. Unfortunately, at this time, the core dump files created from an aborted process using IA-32 execution layer does not contain valid information.

### 8. The I/O Privilege Level (IOPL) mechanism is not implemented

The I/O Privilege Level (IOPL) mechanism is not implemented and is hard coded to 0. As a result, all applications that use the IN or OUT instructions, as well as CLI and STI, will result in a #GP fault.

### 9. Software interrupts must be supported by the OS

Software interrupts (`INT` instructions) are only implemented to the extent that they are supported by the OS, by converting them into an Itanium exception.

### 10. Intersegment calls require OS mechanism

`FAR CALL`, `FAR JMP`, `FAR RET`, `SYSENTER`, and `SYSEXIT` instructions are supported only when there is a standard interface mechanism in the OS. Call gates and hardware task switch mechanisms are not supported.

### 11. Thread creation may be reported incorrectly to the OS

Thread creation may succeed according to the OS, but could later fail inside IA-32 execution layer due to insufficient resources (memory/handle/semaphore). The created thread will never start running.

### 12. Core-dump file may contain Itanium® architecture details

When an IA-32 application running on IA-32 execution layer fails such that it is expected to generate a core-dump file, the generated file reflects Itanium architecture details rather than IA-32 details. Only IA-32 EL Linux versions between 5.3.78 and 5.3.85.34.22 are affected.

### 13. IA-32 process may hang while generating core-dump file

When a multithreaded IA-32 application running on IA-32 execution layer in Linux fails such that it is expected to generate a core-dump file, the process may hang. IA-32 EL Linux versions before 5.3.88.34.22 are affected.

### 14. DLL unload issue

Windows multithreaded applications that perform a dynamic unload of a DLL may receive an Access Violation exception. This may result in the application being unexpectedly terminated. This behavior has only been observed in a synthetic test environment.

§